Title
INCREMENTAL RESTRUCTURING OF RELATIONAL SCHEMAS

Authors
Markowitz, V.M.
Makowsky, J.A.

Publication Date
1987-12-01

Lawrence Berkeley Laboratory
UNIVERSITY OF CALIFORNIA, BERKELEY

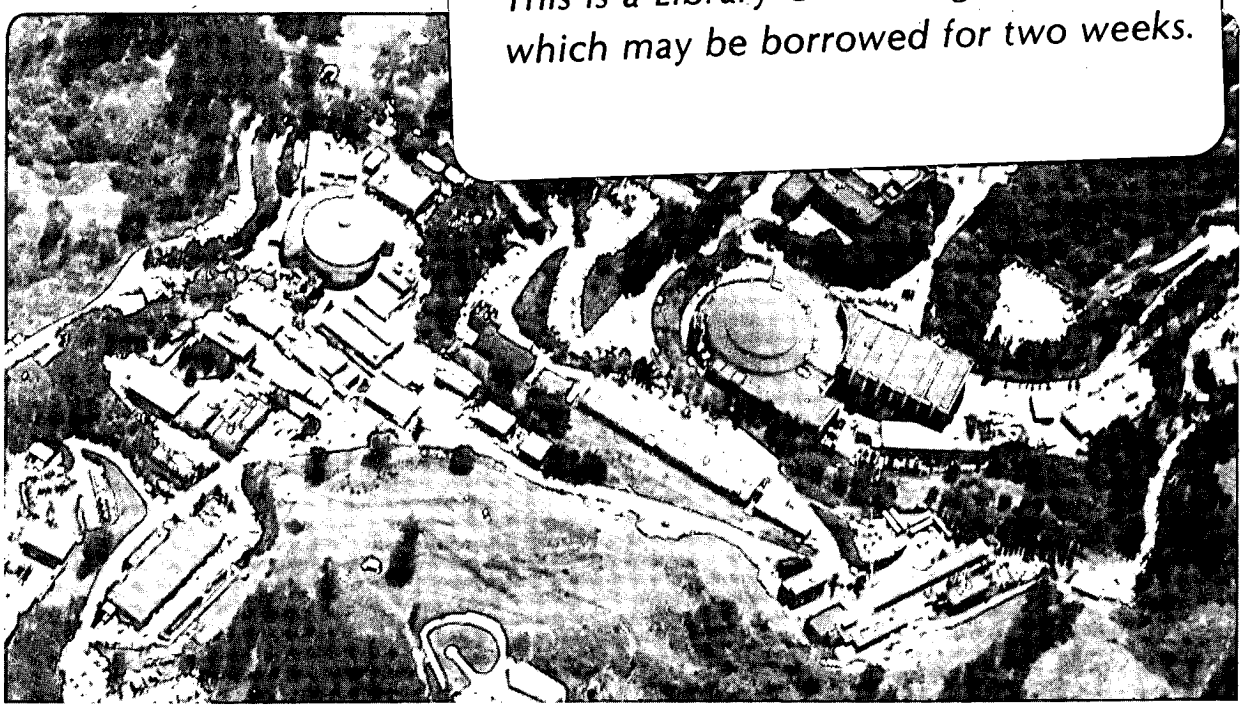
Information and Computing Sciences Division

Presented at the Fourth International Conference on Data Engineering, Los Angeles, CA, February 2–4, 1988

Incremental Restructuring of Relational Schemas

V.M. Markowitz and J.A. Makowsky

December 1987



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

DISCLAIMER

INCREMENTAL RESTRUCTURING OF RELATIONAL SCHEMAS

Victor M. Markowitz
and
Johann A. Makowsky *

Lawrence Berkeley Laboratory
University of California
Computer Science Research Department
Berkeley, California 94720

December 1987

Part of this work has been done while both authors have been with the Computer Science Department of Technion - Israel Institute of Technology, Haifa 32000

* currently on leave from Technion, to the Department of Mathematics, University of Lausanne, Switzerland

LBL is operated under the Department of Energy under contract DE-AC03-76SF00098

INCREMENTAL RESTRUCTURING OF RELATIONAL SCHEMAS

Victor M. Markowitz * and Johann A. Makowsky **

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

** Department of Mathematics, University of Lausanne, Switzerland

Part of this work has been done while both authors have been with the Computer Science Department of Technion - Israel Institute of Technology, Haifa 32000

Abstract. *Schema restructuring* is part of both database *design* and database *reorganization*, which are expressions of the specification and evolution of information systems. *Entity-Relationship (ER) Consistency* expresses the capability of relational databases to model information oriented systems. A relational schema consisting of relation-schemes together with key and inclusion dependencies is said to be ER-consistent if it complies with an entity-relationship structure, meaning that it is representable by an ER-Diagram. *Restructuring* of ER-consistent schemas consists of the addition and removal of relation-schemes, together with the adjustment of key and inclusion dependencies. Smooth schema modification is characterized by *incrementality* and *reversibility*. We propose a *complete* set of incremental and reversible restructuring manipulations for ER-consistent relational schemas. We show how the set of schema restructuring manipulations proposed by us back both *interactive* and *view–integration* methodologies for schema design.

I. INTRODUCTION

Schema restructuring is part of both database design and database reorganization, which are expressions of the specification and evolution of information systems. The relational model fails to provide a suitable framework to deal with information; the relational model user works in terms of data representations which hide most of the structure of the modeled environment. In [8] we have investigated the significance of requiring from a relational database schema to comply with an entity-relationship (ER) structure, that is, to be representable by an ER-Diagram (ERD). Relational database schemas there consist of relation-schemes together with key and inclusion dependencies; such a schema is said to be *Entity-Relationship Consistent (ER-consistent)*, if either it is the translate of, or it is possible to translate it into, an ERD.

The basic *restructuring manipulations* of ER-consistent relational schemas are the addition and removal of relation-schemes together with the suitable modification of key and inclusion dependencies. We define the concepts of *incremental* and *reversible* schema modification, both characterizing smooth schema evolution. While incrementality characterizes the locality of individual schema modifications, reversibility assures that every such modification can be undone in one step. The set of incremental and reversible schema restructuring manipulations proposed by us are translates of *ERD-transformations* consisting of vertex connections and disconnections. We define the *vertex-completeness* of a set of ERD-transformations

as the capability to construct any ERD, such that any vertex connection or disconnection that maps to an incremental and reversible restructuring manipulation, is an atomic transformation. The set of ERD-transformations proposed by us is shown to be vertex-complete. Note that using, different from ours, ERD-transformations in the design of ER-oriented schemas has been proposed in [1], but only in an informal way.

The direct design of relational schemas involving key and inclusion dependencies is a difficult task, mainly because of the excessive power of the inclusion dependencies. Consequently, Sciore [12] has proposed to constrain the set of inclusion dependencies to be *acyclic* and *key-based.* A design methodology pursuing these properties, is presented in [7]. The flatness of the pure relational environment of [7] defeats the declared intention of a simple and natural design. The acyclicity and key basing of the set of inclusion dependencies are captured in a precise manner by ER-consistency. Moreover, ER-oriented schema design makes possible the expression of the *explicit* inclusion dependencies as *inherent* ERD constraints, that is, constraints that are embodied by the schema structure, thus considerably simplifying their use. We show that the *interactive* schema development proposed by [7] can be straightforwardly achieved using our restructuring manipulations.

For complex information systems the design is sometimes split into the design of small (view) schemas subsequently combined into a global schema; this schema design methodology is called *view integration* ([2], [4], [11]). The view integration of [4] is done in a relational environment; its main drawback is its lack of concern in preserving the assumed ER-consistency of the database. In [2] and [11] the view integration is placed in an explicit ER-oriented environment, but *no operations* which would enable a designer to align views for comparison and integration, and to actually perform the integration, are proposed. We show that the restructuring manipulations defined by us fulfill this role.

The paper is organized as follows. The next section of the paper introduces ER Diagrams. In section 3 we review briefly the concept of ER-consistent relational schemas, and investigate the restructuring of ER-consistent schemas. In section 4 we propose a set of ERD transformations, and show that these transformations map to incremental and reversible schema restructuring manipulations. The completeness of this set of ERD-transformations is defined and proved also in section 4. In section 5 we show how our restructuring manipulations can be used in the interactive and view integration methodologies for schema design. We close the paper by drawing some conclusions and outlining directions for further research.

II. ROLE-FREE ENTITY-RELATIONSHIP DIAGRAMS

Entity-Relationship oriented design [5] reflects a natural, although limited, view of the world: entities are qualified by their attributes and interactions between entities are expressed by relationships. An *entity-set* groups entities of a same type, where the entity-type is characterized by the sharing of a same set of attributes. A relationship represents the interaction of several entities, and relationships of the same type are grouped in a *relationship-set.* An attribute is associated with one or several value-sets. Attributes associated with the same collection of value-sets are said to have the same *type.* A subset of the attributes

associated with an entity-set may be specified as the *entity-identifier*, which is used to distinguish among the occurrences of an entity-set. An entity-set in a relationship-set may have a *role*, expressing the function it plays in the relationship-set.

ER-schemas are expressible in a diagrammatic form called *ER-diagram (ERD)* which we define as a directed graph (example in figure 1). Entity-sets, relationship-sets, and attributes of entity-sets or relationship-sets, are represented by entity, relationship and attribute vertices, respectively. Entity, relationship and attribute vertices, are denoted as a-vertices, r-vertices, and e-vertices, respectively, and are represented graphically by circles, diamonds, and rectangles, respectively. ERD vertices are connected by directed edges represented graphically by arrows; edges connecting r-vertices are represented graphically by dashed arrows. Every vertex is labeled by the name of the associated entity-set, relationship-set, or attribute; e-vertices and r-vertices are uniquely identified by their labels globally, while a-vertices are uniquely identified by their labels only locally, within the set of a-vertices connected to some e-vertex/r-vertex. The *reduced ERD* is an ERD with the a-vertices, and all their incident edges, removed. We deal in our paper with ERDs without role specifications, called *role−free ERDs*. A role free ERD does not allow, for instance, the association of entities from a same entity-set. A formal definition of role-freeness is given later (constraint ER3 of definition 2.2). Without loss of generality, we also assume that relationship-sets have no attributes of their own.

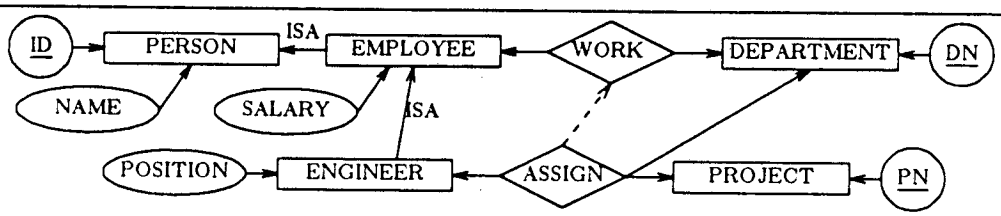
Notations (1):

- A_i , E_i , R_i denote an a-vertex, e-vertex, and r-vertex, resp.;
- $X_i \rightarrow X_j$ denotes an edge between vertices X_i and X_j ;
- $X_i \rightarrow\!\!\rightarrow X_j$ denotes a dipath between vertices X_i and X_j .

ERD *edges* specify *existence constraints:*

$(A_i \rightarrow E_j)$ an attribute does not exist independently, but only related to some entity-set E_j ;

$(E_i \overset{ISA}{\rightarrow} E_j)$ the ISA relationship expresses a *subset* relationship between two entity-sets; E_i is said to be a



Note: ASSIGN → WORK means that an engineer is assigned to projects only in the departments he works in.

Fig.1 Entity-Relationship Diagram Example (*identifiers are underlined*).

subset (specialization) of E_j, and E_j is said to be a *generic entity-set (generalization)* of E_i;

($E_i \overset{ID}{\longrightarrow} E_j$) the ID relationship expresses an identification relationship between an entity-set, called *weak* entity-set, which cannot be identified by its own attributes (E_i), but has to be identified by its relationship(s) with other entity-sets (E_j); E_i is said to be a *dependent* of E_j ;

($R_i \longrightarrow E_j$) relationship-set R_i involves entity-set E_j , therefore a relationship from R_i exists provided the related entity from E_j , also exists;

($R_i \longrightarrow R_j$) a relationship from relationship-set R_i depends on the existence of some relationship from relationship-set R_j .

Notations (2):

- $E_i \overset{ISA}{\longrightarrow\!\!\!\!\rightarrow} E_j$ denotes a dipath of ISA-edges;
- $Atr(E_i) \overset{\Delta}{=} \{A_j \mid A_j \longrightarrow E_i \in G_{ER}\}$, denotes the set of a-vertices connected to an e-vertex E_i ;
- $Id(E_i) \subseteq Atr(E_i)$, denotes the *entity-identifier* specified for e-vertex E_i;
- $GEN(E_i) \overset{\Delta}{=} \{E_k \mid E_i \overset{ISA}{\longrightarrow} E_k \in G_{ER}\}$, denotes the set of generalizations of entity-set E_i;
- $SPEC(E_i) \overset{\Delta}{=} \{E_k \mid E_k \overset{ISA}{\longrightarrow} E_i \in G_{ER}\}$, denotes the set of specializations of entity-set E_i;
- $ENT(E_i) \overset{\Delta}{=} \{E_k \mid E_i \overset{ID}{\longrightarrow} E_k \in G_{ER}\}$, denotes the set of entity-sets on which entity-set E_i is ID-dependent;
- $DEP(E_i) \overset{\Delta}{=} \{E_k \mid E_k \overset{ID}{\longrightarrow} E_i \in G_{ER}\}$, denotes the set of dependents of entity-set E_i ;
- $REL(E_i) \overset{\Delta}{=} \{R_k \mid R_k \longrightarrow E_i \in G_{ER}\}$, denotes the set of relationship-sets involving entity-set E_i ;
- $REL(R_i) \overset{\Delta}{=} \{R_k \mid R_k \longrightarrow R_i \in G_{ER}\}$; denotes the set of relationship-sets depending on relationship-set R_i;
- $DREL(R_i) \overset{\Delta}{=} \{R_k \mid R_i \longrightarrow R_k \in G_{ER}\}$, denotes the set of relationship-sets on which relationship-set R_i depends;
- $ENT(R_i) \overset{\Delta}{=} \{E_k \mid R_i \longrightarrow E_k \in G_{ER}\}$, denotes the set of entity-sets associated by relationship-set R_i ;
- $ENT \longrightarrow\!\!\!\!\rightarrow ENT'$ denotes the existence of an 1-1 correspondence, C , between the e-vertices of two sets of e-vertices, ENT and ENT', belonging to an ERD, G_{ER} :

$$C = \{ (E_i , E_j) \mid E_i \in ENT , E_j \in ENT' \text{ and } (\text{ either } E_i \longrightarrow\!\!\!\!\rightarrow E_j \in G_{ER} \text{ or } E_i \equiv E_j) \}.$$

Definition 2.1 - *Specialization Cluster.*

Let G_{ER} be an ERD, and $E_i \in G_{ER}$, an e-vertex; the *specialization cluster rooted* in E_i, $SPEC^*(E_i)$, is the set of all the e-vertices representing *specializations* of the entity-set represented by E_i :

$$SPEC^*(E_i) = E_i \cup \{E_j \mid E_j \overset{ISA}{\longrightarrow\!\!\!\!\rightarrow} E_i \in G_{ER}\}.$$

If E_i has no generalization, that is, $GEN(E_i) = \varnothing$, then the specialization cluster is said to be *maximal*. In figure 1, for instance, $SPEC^*$(PERSON) is {PERSON, EMPLOYEE, ENGINEER}, and it is maximal.

Definition 2.2 - *Role-Free ER-Diagram.*

A *Role-Free ERD* is a finite labeled digraph $G_{ER}=(V, H)$, where

V is the disjoint union of three subsets of vertices: E (e-vertices), R (r-vertices), and A (a-vertices); H is the set of directed edges, where an edge can be of one of the following forms: $A_i \rightarrow E_j$, $E_i \rightarrow E_j$, $R_i \rightarrow E_j$, and $R_i \rightarrow R_j$.

G_{ER} obeys the following constraints:

(ER1) G_{ER} is an acyclic digraph without parallel edges;

(ER2) $\forall\, A_i \in G_{ER}$: $outdegree(A_i) = 1$;

(ER3) for any e-vertex/r-vertex X_i holds:

$$\forall\, (E_j\,,\, E_k) \in ENT(X_i) \times ENT(X_i) :\ uplink(\, E_j\,,\, E_k\,) = \varnothing\ ;$$

(uplink is defined below - definition 2.3)

(ER4) $\forall\, E_i \in G_{ER}$: if $GEN(E_i) \neq \varnothing$ then $Id(E_i)=\varnothing$; $ENT(E_i)=\varnothing$;

and E_i belongs to a ***unique maximal specialization cluster***;

otherwise $Id(E_i) \neq \varnothing$;

(ER5) $\forall\, R_i \in G_{ER}$: $ENT(R_i) \geq 2$ and $\forall\, R_i \rightarrow R_j \in G_{ER}$:

$$\exists\, ENT \subseteq ENT(R_i) \text{ such that } ENT \rightarrow\!\rightarrow ENT(R_j).$$

Constraint (ER1) above guarantees that directed cycles do not exist so that, for instance, an entity-set will neither be defined as depending on identification on itself, nor be defined as a proper subset of itself. An attribute characterizes a single entity-set, therefore constraint (ER2). Constraint (ER3) states the role-freeness condition; it assures, additionally, the uniqueness of the correspondence of two related relationship-sets (ER5). The rules of identifier specification are given by constraint (ER4); (ER4) also states that every generalization hierarchy is a rooted tree.

Definition 2.3 - *Uplink.*

Let $G_{ER} = (\, V\,,\, H\,)$ be an ERD, and E_i an e-vertex of G_{ER}. E_i is said to be an *upper link (uplink)* of the e-vertex set $\Lambda = \{E_j \mid E_j \in G_{ER}\}$, iff $\forall\, E_j \in \Lambda$: $E_j \rightarrow\!\rightarrow E_i \in G_{ER}$ (possibly of length 0), and there is no E_k ($k \neq i$) , such that $E_k \rightarrow\!\rightarrow E_i \in G_{ER}$ and $\forall\, E_j \in \Lambda$: $E_j \rightarrow\!\rightarrow E_k \in G_{ER}$. The set of all *uplinks* of a set of e-vertices Λ , is denoted $uplink(\Lambda)$.

In figure 1, for instance, *uplink*(ENGINEER, EMPLOYEE) is {EMPLOYEE}.

Definition 2.4 - *ER-Compatibility, Quasi-Compatibility.*

The entity-set, relationship-set and attribute compatibility, have the following graph-oriented analogs: (i) two a-vertices, A_i and A_j, are said to be *ER-compatible* iff they have the same type; (ii) two e-vertices, E_i and E_j, are said to be *ER-compatible* iff they belong to a same specialization cluster; and *quasi-compatible* iff their identifiers are compatible and $ENT(E_i) \equiv ENT(E_j)$; and (iii) two r-vertices, R_i and

R_j, are said to be *ER-compatible* iff there is a one-to-one correspondence, $Comp(R_i,R_j)$, of compatible e-vertices between $ENT(R_i)$ and $ENT(R_j)$: $Comp(R_i,R_j)=\{(E_k,E_m) \mid E_k \in ENT(R_i), E_m \in ENT(R_j)$, and E_k and E_m are compatible$\}$; the role-freeness assures the *uniqueness* of this correspondence, whenever it exists.

The entity-set, relationship-set and attribute compatibility, have a straightforward intuition, while entity-set quasi-compatibility expresses the capability of *generalization* of the respective entity-sets.

III. RESTRUCTURING OF ER-CONSISTENT SCHEMAS

A *relational schema* is a pair (R, D) where R is a set of relation schemes, $R = (R_1,...,R_k)$, and D is a set of dependencies over R. We deal with two kinds of dependencies, one inner relational, and one inter relational, defined below. A *relation-scheme* is a named set of attributes, $R_i(A_i)$. On the semantic level, every attribute is assigned a domain (the relational correspondent of the ER value-set). A *database state* of R is defined as $r = <D_1,...,D_m,r_1...r_k>$, where r_i is assigned a subset of the cartesian product of the domains corresponding to its attributes. Provided the domains are sets of *interpreted* values which are restricted conceptually and operationally, two attributes are said to be *compatible* if they are associated with a same domain. In the following definition R denotes a set of relation-schemes and $R_i \in R$.

Definition 3.1 - *Functional Dependency, Key, Key Graph.*

(i) *functional dependency* (FD) over $R_i(A_i)$ is a statement of the form $X \rightarrow Y$, where $X \subseteq A_i$ and $Y \subseteq A_i$; $X \rightarrow Y$ is valid in a state r iff for any two tuples of r_i, t and t', $t[X]=t'[X]$ implies $t[Y]=t'[Y]$;

(ii) *key dependency* over $R_i(A_i)$, is an FD $K_i \rightarrow A_i$, where $K_i \subseteq A_i$; K_i is called *key;* note that keys need not be minimal, that is, K_i is a key even if there is a strict subset of K_i which is also a key;

(iii) *correlation key* of R_i, CK_i, is the union of all the subsets of A_i, that appear as keys in some relation R_j, $j \neq i$;

(iv) *key graph* associated with R, is a digraph $G_K=(V, E)$, where $V=R$ and $R_i \rightarrow R_j \in E$ iff (i) $CK_i = K_j$; or (ii) $K_j \subsetneq CK_i$ and $\nexists R_k$ such that $K_j \subsetneq CK_k$ and $K_k \subsetneq CK_i$.

Definition 3.2 - *Inclusion Dependency, Properties, Graph.*

(i) *inclusion dependency* (IND) is a statement of the form $R_i[X] \subseteq R_j[Y]$, where X and Y are subsets of A_i and A_j, respectively, and $|X| = |Y|$; an IND $R_i[X] \subseteq R_j[Y]$, is valid in a state r, iff $r_i[X] \subseteq r_j[Y]$;

(ii) $R_i[X] \subseteq R_j[Y]$, is said to be *typed* [4] iff $X = Y$;

(iii) $R_i[X] \subseteq R_j[Y]$, is said to be *key-based* [12] iff $Y = K_j$;

(iv) for a set of inclusion dependencies, I, over R, the associated *IND graph* is the digraph $G_I=(V, E)$, where $V = R$, and $R_i \rightarrow R_j \in E$ iff $R_i[X] \subseteq R_j[Y] \in I$;

(v) a set of inclusion dependencies, I, is said to be *cyclic* if either $R_i[X_i] \subseteq R_i[Y_i]$ for $X \neq Y$, or there are $R_1 \ldots R_n$ such that $R_i[X_i] \subseteq R_1[Y_1]$, $R_1[X_1] \subseteq R_2[Y_2]$, ..., $R_n[X_n] \subseteq R_i[Y_i]$; a set of inclusion dependencies, I, is *acyclic* iff the associated IND graph is an acyclic digraph [12].

The sets of keys and inclusion dependencies associated with some relational schema, are denoted K and I, respectively.

Proposition 3.1 (Theorem 5.1 [4]). Given a set of typed inclusion dependencies, I, every inclusion dependency $R_i[X] \subseteq R_j[Y]$ is implied by I iff either it is trivial, or $X=Y$, and there is a path from R_i to R_j in the associated IND graph, corresponding to a sequence of INDs of I, $R_i[W] \subseteq \cdots \subseteq R_j[W]$, such that $X \subseteq W$.

Proposition 3.2 (Theorem 5.3 [4]). Let I and K be sets of inclusion dependencies and keys, respectively in a relational schema (R,K,I), such that I is key based. Then $(I \cup K)^+ = I^+ \cup K^+$.

In [8] and [9] we have proposed the ERD as a higher-level schema for the relational model. The relational interpretation of an ERD is given by its mapping into a relational schema. A relational schema which is the translate of an ERD, is said to be (trivially) *ER-consistent*. Then a *state* of an ERD is the state of its relational translate. A relational database whose schema is ER-consistent, is said to be an *ER-consistent database*. In [9] we have presented the direct mapping (figure 2) and reverse mapping between role-free ERDs and relational schemas of the form (R, K, I). We briefly review bellow some results of [9] (presented partially in [8]).

Proposition 3.3 (Proposition 4.1 [9]). Let (R, K, I) be an ER-consistent relational schema, the translate of the ERD G_{ER}, whose reduced ERD is G'_{ER}, and let G_I and G_K be the inclusion dependency and key graphs associated with (R, K, I), respectively. (i) G_I and G'_{ER} are isomorphic; (ii) I is typed, key-based, and acyclic; and (iii) G_I is a subgraph of G_K.

Proposition 3.4 (Corollary 4.2 [9]). Let (R, K, I) be an ER-consistent relational schema; an inclusion dependency $R_i[X] \subseteq R_j[Y]$ is implied by I iff either it is trivial, or $X=Y$ and there is a path from R_i to R_j in the associated IND graph.

Notation: typing and key-basing allow to denote an inclusion dependency of an ER-consistent database, $R_i[K_j] \subseteq R_j[K_j]$, as $R_i \subseteq R_j$.

Schema restructuring is part of both database design and database reorganization. For ER-consistent relational schemas, the restructuring manipulations are the addition and removal of relation schemes, together with the addition and removal of the involved key and inclusion dependencies. Since adding and removing relations are expressions of information-structure specification and evolution, ER-

Input: $G_{ER}=(V, H)$, an ERD;

Output: the relational schema (R, K, I) interpreting G_{ER};

(1) prefix the labels of the a-vertices belonging to entity-identifiers by the label of the corresponding e-vertex;

(2) for every e-vertex/r-vertex X_i define recursively the following set of a-vertices:

$$Key(X_i) = Id(X_i) \bigcup_{X_i \rightarrow X_j \in G_{ER}} Key(X_j) \ ;$$

(3) for every e-vertex/r-vertex X_i: define relation-scheme R_i;

$$K_i := Key(X_i); \qquad A_i := Atr(X_i) \cup Key(X_i);$$
$$K := K \cup K_i; \qquad R := R \cup R_i(A_i);$$

(4) let R_i and R_j be two relation schemes corresponding to e-vertices/r-vertices X_i and X_j, respectively; for every edge $X_i \rightarrow X_j \in G_{ER}$: $I := I \cup (R_i[K_j] \subseteq R_j[K_j])$.

Fig.2 $\mathbf{T_e}$: Mapping ER-Diagram Into Relational Schema.

consistent relational schemas are well suited for defining schema restructuring manipulations. We assume in this paper that the database state is *empty*. The coupling of schema restructuring manipulations with state mappings is investigated in [10].

Definition 3.3 - *Relation-Scheme Addition and Removal.*
Let (R , K , I) be an ER-consistent relational schema, and R_i a relation-scheme. The *addition/removal* of R_i , denoted σ_i , maps (R , K , I) to (R', K', I') , such that

addition $R' := R \cup R_i$, $K' := K \cup K_i$, and $I' := I \cup I_i - I_i^+$,

where $I_i = \{R_j \subseteq R_i \mid R_j \in R\} \cup \{R_i \subseteq R_j \mid R_j \in R\}$,

s.t. for any pair $R_j \subseteq R_i$, $R_i \subseteq R_k$ of $I_i : R_j \subseteq R_k \in I^+$,

$I_i^+ = \{R_j \subseteq R_k \mid R_j \subseteq R_k \in I, R_j \subseteq R_i \in I_i, R_i \subseteq R_k \in I_i\}$;

removal $R' := R - R_i$, $K' := K - K_i$, and $I' := I - I_i \cup I_i^+$,

where $I_i = \{R_j \subseteq R_i \mid R_j \subseteq R_i \in I\} \cup \{R_i \subseteq R_j \mid R_i \subseteq R_j \in I\}$,

$I_i^+ = \{R_j \subseteq R_k \mid R_j \subseteq R_k \notin I, R_j \subseteq R_i \in I, R_i \subseteq R_k \in I\}$.

Smooth schema restructuring, without major disruptions, is characterized by incrementality, defined below. Informally, incrementality requires from a single manipulation to affect only *locally* the schema,

that is, to keep invariant the schema segment which is not in the immediate neighborhood of the manipulation. Accordingly, the effects of every single manipulation are easy to comprehend and manage. While incrementality characterizes one-step schema modifications, reversibility assures that every such modification can be undone in one step .

Definition 3.4 *-Incremental and Reversible Schema Restructuring.*

Let (R, K, I) be a relational schema mapped to (R', K', I') by an addition/removal restructuring manipulation, and let I_i be the subset of inclusion dependencies involving relation scheme R_i. A restructuring manipulation is said to be

(i) *incremental* iff either

$[\underline{\text{addition}} \text{ of } R_i]: R' = R \cup R_i,\ K' = K \cup K_i$, and $(I' \cup K')^+ = (I \cup K \cup I_i \cup K_i)^+$; or

$[\underline{\text{removal}} \text{ of } R_i]: R' = R - R_i,\ K' = K - K_i$, and $(I' \cup K')^+ = ((I \cup K)^+ - I_i - K_i)^+$;

(ii) *reversible* iff there is another restructuring manipulation such that their sequence applied on (R, K, I), returns the same schema, up to a renaming of attributes.

Note that verifying incrementality for unrestricted relational schemas, might be exponential, or even undecidable (for the complexity of the implication problem for inclusion and functional dependencies see [3]), while for ER-consistent schemas the verification is polynomial (propositions 3.2 and 3.4).

Proposition 3.5. The restructuring manipulations of ER-consistent relational schemas are incremental and reversible.

Sketch of Proof: straightforward, based on propositions 3.2 and 3.4.

IV. ENTITY-RELATIONSHIP DIAGRAM TRANSFORMATIONS

The major problems with the restructuring of ER-consistent relational schemas are the preservation of the ER-consistency of the schema, and the specification of ER-oriented correspondents for the restructuring manipulations. We choose to deal with these problems by first defining a set of *ERD-transformations*, Δ, consisting of connections and disconnections of vertices. Next, we shall specify the mapping of ERD-transformations into restructuring manipulations that we shall show to be incremental and reversible, and we shall prove the completeness of the set Δ.

The simplest ERD-transformations are the connection and disconnection of attribute-vertices; they have obvious prerequisites and ERD mappings, and they have the following form: *Connect* / *Disconnect* A_i *to* / *from* E_j where A_i and E_j denote an a-vertex and e-vertex, respectively. Because of the dependence of an attribute on the entity-set it characterizes, the connection or disconnection of a-vertices appears only embedded in other transformations. In order to simplify the definitions we shall omit the specification of attributes that do not belong to entity-identifiers, whenever the extension of the respective definition is obvious.

We partition the set Δ of ERD-transformations into three classes: (Δ_1) connection and disconnection of vertices representing entity-subsets and relationship-sets; (Δ_2) connection and disconnection of vertices representing entity-sets without dependent entity-sets, or representing generalizations of other entity-sets; (Δ_3) connection and disconnection of vertices representing conversions of other vertices.

4.1. ERD-Transformations : Class Δ_1

The class Δ_1 of ERD transformations consists of connections and disconnections of vertices representing entity-subsets and relationship-sets. An example is given in figure 3.

4.1.1. Connect/Disconnect Entity-Subset. A new entity-set, necessarily with an empty identifier, can be specified as the *specialization* of several ER-compatible entity-sets (GEN below); additionally, it can be specified as the generalization of ER-compatible entity-sets ($SPEC$ below), it can have dependent entity-sets (DEP below), and it can be involved in relationship-sets (REL below).

Syntax: Connect E_i isa GEN [gen $SPEC$] [inv REL] [det DEP]

where E_i denotes an e-vertex, GEN, $SPEC$, and DEP, denote sets of e-vertices, and REL denotes a set of r-vertices.

Prerequisites:

(i) $E_i \notin G_{ER}$, $GEN \neq \emptyset$, $\forall E_j \in (GEN \cup SPEC)$: $E_j \in G_{ER}$;

(ii) neither GEN nor $SPEC$ include e-vertices connected by directed paths in G_{ER} ;

(iii) $GEN \cup SPEC$ is a set of ER-compatible e-vertices, and

$$\forall (E_k, E_j) \in SPEC \times GEN : E_k \overset{ISA}{\longrightarrow} E_j \in G_{ER} ;$$

(iv) $\forall R_k \in REL : \exists E_j \in GEN$ s.t. $R_k \rightarrow E_j \in G_{ER}$;

(v) $\forall E_k \in DEP : \exists E_j \in GEN$ s.t. $E_k \overset{ID}{\rightarrow} E_j \in G_{ER}$.

G_{ER} *mapping* : add E_i;

add-edge $\{E_i \overset{ISA}{\rightarrow} E_j \mid E_j \in GEN\}, \{E_j \overset{ISA}{\rightarrow} E_i \mid E_j \in SPEC\},$
$\{R_k \rightarrow E_i \mid R_k \in REL\}, \{E_k \overset{ID}{\rightarrow} E_i \mid E_k \in DEP\};$

remove-edge $\{E_k \overset{ISA}{\rightarrow} E_j \mid E_k \in SPEC, E_j \in GEN\}, \{R_k \rightarrow E_j \mid R_k \in REL, E_j \in GEN\},$
$\{E_k \overset{ID}{\rightarrow} E_j \mid E_k \in DEP, E_j \in GEN\}.$

The disconnection of an entity-subset is straightforward except the case when it has dependent entity-sets, or when it is involved in relationship-sets: then it is necessary to specify the distribution of all its dependents and relationship-sets among its generalization entity-sets ($XDEP$ and $XREL$, respectively, below).

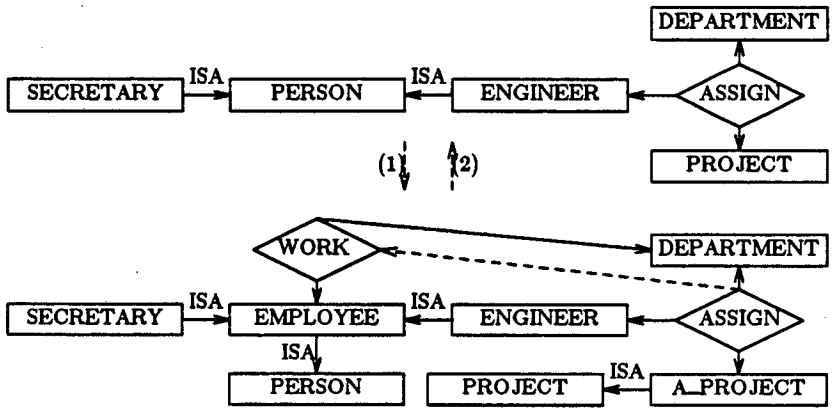


Fig.3 (1) *Connect* EMPLOYEE *isa* PERSON *gen* {SECRETARY, ENGINEER}
Connect A_PROJECT *isa* PROJECT *inv* ASSIGN
Connect WORK *rel* {EMPLOYEE, DEPARTMENT} *det* ASSIGN
(2) *Disconnect* WORK ; A_PROJECT; EMPLOYEE.

Syntax: *Disconnect* E_i [*dis* *XREL*] [*dis* *XDEP*]

where E_i denotes an e-vertex, *XREL* denotes a set of pairs of r-vertices and e-vertices, (R_j, E_k) , and *XDEP* denotes a set of pairs of e-vertices, (E_j, E_k) .

Prerequisites:

(i) $E_i \in G_{ER}$, $GEN(E_i) \neq \emptyset$;

(ii) $\{R_k \mid R_k$ appears in *XREL* $\}=REL(E_i)$, $\forall (R_k, E_j) \in XREL : E_j \in GEN(E_i)$;

(iii) $\{E_k \mid E_k$ appears in *XDEP* $\}=DEP(E_i)$, $\forall (E_k, E_j) \in XDEP : E_j \in GEN(E_i)$.

G_{ER} *mapping* : remove E_i;

add-edge $\{E_j \overset{ISA}{\longrightarrow} E_k \mid E_j \in SPEC(E_i), E_i \in GEN(E_i), E_j \overset{ISA}{\longrightarrow} E_k \notin G_{ER}\}$,

$\{R_j \longrightarrow E_k \mid (R_j, E_k) \in XREL\}$, $\{E_j \overset{ID}{\longrightarrow} E_k \mid (E_j, E_k) \in XDEP\}$;

remove-edge $\{E_j \overset{ISA}{\longrightarrow} E_i \mid E_j \in SPEC(E_i)\}$, $\{E_i \overset{ISA}{\longrightarrow} E_k \mid E_k \in GEN(E_i)\}$,

$\{R_j \longrightarrow E_i \mid R_j \in REL(E_i)\}$, $\{E_j \overset{ID}{\longrightarrow} E_i \mid E_j \in DEP(E_i)\}$.

4.1.2. Connect/Disconnect Relationship-Set. A new relationship-set can be specified as associating existing entity-sets (*ENT* below); it could depend on other relationship-sets (*DREL* below), and other relationship-sets could depend on it (*REL* below).

Syntax: *Connect* R_i *rel* *ENT* [*dep* *DREL*] [*det* *REL*]

where R_i denotes an r-vertex, ENT denotes a set of e-vertices, and REL, $DREL$ denote sets of r-vertices.

Prerequisites:

(i) $R_i \notin G_{ER}$, $\forall E_j \in ENT$: $E_j \in G_{ER}$, $\forall R_j \in (REL \cup DREL)$: $R_j \in G_{ER}$;

(ii) $|ENT| \geq 2$, and $\forall E_j, E_k \in ENT$: $uplink(E_j, E_k) = \emptyset$;

(iii) neither REL, nor $DREL$, include r-vertices connected by directed paths in G_{ER} ;

(iv) $\forall (R_k, R_j) \in REL \times DREL$: $R_k \rightarrow\!\!\!\rightarrow R_j \in G_{ER}$;

(v) $\forall R_k \in REL$: $\exists ENT' \subseteq ENT(R_k)$ s.t. $ENT' \hookrightarrow\!\!\!\rightarrow ENT$;

(vi) $\forall R_j \in DREL$: $\exists ENT' \subseteq ENT$, s.t. $ENT' \hookrightarrow\!\!\!\rightarrow ENT(R_j)$.

G_{ER} *mapping* : add R_i;

add-edge $\{R_i \rightarrow E_j \mid E_j \in ENT\}, \{R_i \rightarrow R_j \mid R_j \in DREL\}, \{R_k \rightarrow R_i \mid R_k \in REL\}$;

remove-edge $\{R_k \rightarrow R_j \mid R_k \in REL, R_j \in DREL\}$.

The disconnection of a relationship-set is straightforward.

Syntax: *Disconnect* R_i where R_i denotes an r-vertex.

Prerequisites: $R_i \in G_{ER}$.

G_{ER} *mapping* : remove R_i;

add-edge $\{R_j \rightarrow R_k \mid R_j \in REL(R_i), R_k \in DREL(R_i), R_j \rightarrow R_k \notin G_{ER}\}$;

remove-edge $\{R_j \rightarrow R_i \mid R_j \in REL(R_i)\}, \{R_i \rightarrow R_k \mid R_k \in DREL(R_i)\}, \{R_i \rightarrow E_k \mid E_k \in ENT(R_i)\}$.

4.2. ERD-Transformations : Class Δ_2

The class Δ_2 of ERD-transformations consists of connections and disconnections of vertices representing entity-sets without dependent entity-sets, possibly representing generalizations of other entity-sets. The relation to existing entity-sets defines whether the respective entity-set is an independent, weak, or generic entity-set. An example is given in figure 4.

4.2.1 Connect/Disconnect Independent/Weak Entity-Set. A new *independent/weak* entity-set must have a non-empty identifier and it no dependent entity-sets; a weak entity-set depends for identification on other entity-sets (ENT below).

Syntax: *Connect* $E_i(Id_i)$ [*id* ENT]

where ENT denotes a set of e-vertices, E_i denotes an e-vertex, and Id_i denotes a set of identifier a-vertices.

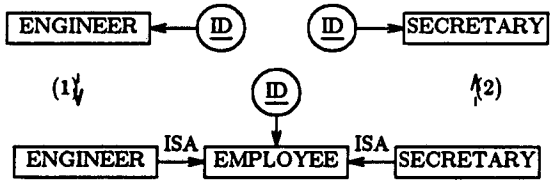


Fig.4 (1) *Connect* EMPLOYEE(ID) *gen* {ENGINEER, SECRETARY}
(2) *Disconnect* EMPLOYEE

Prerequisites:

(i) $E_i \notin G_{ER}$, and $\forall A_j \in Id_i : A_j \notin G_{ER}$;

(ii) $\forall E_j , \forall E_k \in ENT : \ uplink(E_j , E_k) = \varnothing$.

G_{ER} *mapping* : add E_i;

add-edge $\{E_i \overset{ID}{\rightarrow} E_j \mid E_j \in ENT\}$;

connect $\{Connect\ A_j\ to\ E_i \mid A_j \in Id_i\}$.

The disconnection of an independent/weak entity-set is *prohibited* either when it has dependent entity-sets, or when it is involved in relationship-sets (then the dependent entity-sets/relationship-sets must be first removed).

Syntax: *Disconnect* E_i where E_i denotes an e-vertex.

Prerequisites:

$E_i \in G_{ER}$, $SPEC(E_i) = \varnothing$, $REL(E_i) = \varnothing$, $DEP(E_i) = \varnothing$.

G_{ER} *mapping* : remove E_i;

remove-edge $\{E_i \overset{ID}{\rightarrow} E_k \mid E_k \in ENT(E_i)\}$.

4.2.2. Connect/Disconnect Generic Entity-Set. A new entity-set with a non empty identifier can be defined as the *generalization* of several *quasi-compatible* entity-sets (*SPEC* below).

Syntax: *Connect* $E_i(Id_i)$ *gen* *SPEC*

where E_i denotes an e-vertex, Id_i denotes a set of a-vertices, and *SPEC* denotes a set of e-vertices; it must exist a *compatibility correspondence* between Id_i and the entity-identifier of every e-vertex of *SPEC* – this correspondence defines the value-set association (type) of the a-vertices of Id_i .

Notations: $ENT \stackrel{\Delta}{=} \{E_k \mid E_j \stackrel{ID}{\rightarrow} E_k \in G_{ER}\ ,\ E_j \in SPEC\}$,
common to all the e-vertices of $SPEC$.

Prerequisites:

(i) $E_i \notin G_{ER}$, $\forall E_j \in SPEC : E_j \in G_{ER}$ and $\mid Id(E_j) \mid = \mid Id_i \mid$;
(ii) $\forall E_k, E_j \in SPEC : E_k$ and E_j are *quasi–compatible*.

G_{ER} *mapping* : add E_i;

add-edge $\{E_j \stackrel{ISA}{\rightarrow} E_i \mid E_j \in SPEC\}, \{E_i \stackrel{ID}{\rightarrow} E_k \mid E_k \in ENT\}$;

connect $\{Connect\ A_j\ to\ E_i \mid A_j \in Id_i\}$;

remove-edge $\{E_j \stackrel{ID}{\rightarrow} E_k \mid E_j \in SPEC\ ,\ E_k \in ENT\}$;

disconnect $\{Disconnect\ A_j\ from\ E_k \mid E_k \in SPEC, A_j \in Id(E_k)\}$.

The disconnection of a generic entity-set causes the *distribution* of its identifier attributes among its specialization entity-sets. The disconnection of an entity-set is *prohibited* either when it might split specialization clusters, when it has dependent entity-sets, or when it is involved in relationship-sets (then the dependent entity-sets/relationship-sets must be first removed).

Syntax: *Disconnect* E_i where E_i denotes an e-vertex.

Prerequisites:

(i) $E_i \in G_{ER}$, $GEN(E_i) = \emptyset$, $REL(E_i) = \emptyset$, $DEP(E_i) = \emptyset$;
(ii) $SPEC(E_i) \neq \emptyset$, and $\forall E_k, E_j \in SPEC(E_i) : SPEC^*(E_k) \cap SPEC^*(E_j) = \emptyset$.

G_{ER} *mapping* : remove E_i;

distribute $\{Connect\ A'_k\ to\ E_j \mid A'_k$ is a duplicate of $A_k \in Id(E_i)$, and $E_j \in SPEC(E_i)\}$;

add-edge $\{E_j \stackrel{ID}{\rightarrow} E_k \mid E_j \in SPEC(E_i)\ ,\ E_k \in ENT(E_i)\}$;

remove-edge $\{E_j \stackrel{ISA}{\rightarrow} E_i \mid E_j \in SPEC(E_i)\}, \{E_i \stackrel{ID}{\rightarrow} E_k \mid E_k \in ENT(E_i)\}$.

Note that the generic entity-set connection and disconnection can be straightforwardly extended to include the unification, respectively the distribution, of compatible non-identifier attributes.

4.3. ERD-Transformations : Class Δ_3

In the real-world there is no clear-cut separation between entities, relationships and attributes. The same information can be perceived differently, either by different users, or as part of the information-system evolution; the ability to view the same information differently in various contexts, is captured by the database modeling concept of *semantic relativism*. The class Δ_3 of ERD transformations consist of *conversion* transformations, which allow to change identifier attributes into weak entity-sets, and weak entity-sets into independent entity-sets, together with their reverse conversions. Examples for these transformations are given in figures 5 and 6.

4.3.1. Conversion of Identifier-Attributes into Weak Entity-Set. An entity can be perceived as the aggregation of the values of its attributes; by splitting this aggregation new, weak, entities can be specified. The conversion of a set of attributes into a weak entity-set is performed by the connection of an entity-vertex.

Syntax: *Connect* $E_i(Id_i, Atr_i)$ *con* $E_j(Id_j, Atr_j)$ [*id* ENT]

where E_i and E_j denote e-vertices, ENT denote a set of e-vertices, Id_i, Id_j, Atr_i, and Atr_j denote sets of a-vertices; it must exist a *compatibility correspondence* between Id_i and Id_j, and Atr_i and Atr_j, respectively- this correspondence defines the value-set association (type) of the a-vertices of Id_i and Atr_i; the conversion refers *only* to identifier attributes.

Prerequisites:

(i) $E_i \notin G_{ER}$, $\forall A_k \in Id_i \cup Atr_i$: $A_k \notin G_{ER}$;

(ii) $E_j \in G_{ER}$, $Id_j \subsetneq Id(E_j)$, $Atr_j \subseteq Atr(E_j) - Id(E_j)$, $ENT \subseteq ENT(E_j)$;

(iii) $| Id_i | = | Id_j |$ and $| Atr_i | = | Atr_j |$.

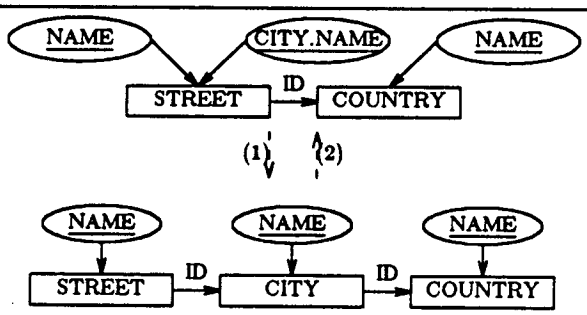


Fig.5 (1) *Connect* CITY(NAME) *con* STREET(CITY.NAME) *id* COUNTRY
(2) *Disconnect* CITY(NAME) *con* STREET(CITY.NAME)

G_{ER} *mapping* : add E_i;

connect $\{\underline{Connect}\ A_k\ \underline{to}\ E_i \mid A_k \in Id_i \cup Atr_i\}$;

disconnect $\{\underline{Disconnect}\ A_k\ \underline{from}\ E_j \mid A_k \in Id_j \cup Atr_j\}$;

add-edge $E_j \overset{ID}{\longrightarrow} E_i$, $\{E_i \overset{ID}{\longrightarrow} E_k \mid E_k \in ENT\}$;

remove-edge $\{E_j \overset{ID}{\longrightarrow} E_k \mid E_k \in ENT\}$.

The reverse transformation is the conversion of a weak entity-set into one or several identifier attributes. The disconnection-conversion of a weak entity-set is *prohibited* when it has specialization entity-sets or is involved in relationship-sets.

Syntax: $\underline{Disconnect}\ E_i(Id_i\ ,\ Atr_i)\ \ \underline{con}\ E_j(Id_j\ ,\ Atr_j)$

where E_i and E_j, Id_i, Id_j, Atr_i , and Atr_j, are as above, with the compatibility correspondence now defining the value-set association (type) of the a-vertices of Id_j and Atr_j .

Prerequisites:

(i) $E_i \in G_{ER}$, $DEP(E_i) = \{E_j\}$, $SPEC(E_i) = \emptyset$, $REL(E_i) = \emptyset$;

(ii) $Id_i = Id(E_i)$, and $Atr_i = Atr(E_i) - Id(E_i)$;

(iii) $\forall\ A_k \in (Id_j \cup Atr_j)$: $A_k \notin G_{ER}$, $\mid Id_j \mid = \mid Id_i \mid$, $\mid Atr_j \mid = \mid Atr_i \mid$.

G_{ER} *mapping* : remove E_i;

connect $\{\underline{Connect}\ A_k\ \underline{to}\ E_j \mid A_k \in Id_j \cup Atr_j\}$;

add-edge $\{E_j \overset{ID}{\longrightarrow} E_k \mid E_k \in ENT(E_i)\}$;

remove-edge $E_j \overset{ID}{\longrightarrow} E_i$, $\{E_i \overset{ID}{\longrightarrow} E_k \mid E_k \in ENT(E_i)\}$.

The above two ERD-transformations can be straightforwardly extended to non identifier attributes, provided the relational database allows the use of nulls and supports the definition of multivalued attributes (one-level nested relations [6]).

4.3.2. Conversion of Weak into Independent Entity-Set. A weak entity-set embeds the association of its entities with the entities their existence depend on; it can be perceived as an independent entity-set, associated with other entity-sets through a *stand-alone* relationship-set. This conversion is performed by dis-embedding the corresponding relationship-set, that is, by converting the weak entity-set into a relationship-set, and by specifying an additional independent entity-set. The conversion of a weak entity-set is *prohibited* when it has dependent or specialization entity-sets.

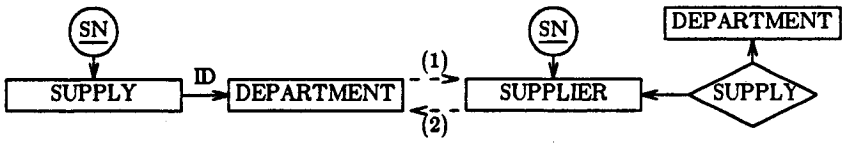


Fig.6 (1) *Connect* SUPPLIER *con* SUPPLY
(2) *Disconnect* SUPPLIER *con* SUPPLY

Syntax: *Connect* E_i *con* E_j where E_i and E_j denote e-vertices.

Prerequisites:

$E_i \notin G_{ER}$, $E_j \in G_{ER}$, $ENT(E_j) \neq \emptyset$, $DEP(E_j) = \emptyset$, $SPEC(E_j) = \emptyset$, $REL(E_j) = \emptyset$.

G_{ER} *mapping* : add E_i, convert E_j into R_j ;

add-edge $R_j \rightarrow E_i$.

The reverse transformation is the conversion of an independent entity-set into a weak entity-set, by embedding the entity-set into the necessarily ***unique*** relationship-set involving it; the conversion is performed by removing the independent entity-set and by converting the respective relationship-set into a weak entity-set. The conversion of an independent entity-set is ***prohibited*** when it has dependent or specialization entity-sets, or when the involved relationship-set depends on other relationship-sets.

Syntax: *Disconnect* E_i *con* R_j

where E_i denotes an e-vertex, and R_j denotes an r-vertex.

Prerequisites:

(i) $E_i \in G_{ER}$, $DEP(E_i) = \emptyset$, $SPEC(E_i) = \emptyset$;

(ii) $REL(E_i) = \{R_j\}$, $REL(R_j) = \emptyset$, $DREL(R_j) = \emptyset$.

G_{ER} *mapping* : remove E_i, convert R_j into E_j ;

add-edge $\{ E_j \overset{ID}{\rightarrow} E_k \mid E_k \in ENT(E_i) \}$, and

remove-edge $E_j \rightarrow E_i$, $\{E_i \overset{ID}{\rightarrow} E_k \mid E_k \in ENT(E_i)\}$.

Proposition 4.1. Let G_{ER} be an ERD, and τ_i an Δ-transformation. Then τ_i maps correctly G_{ER}.
Sketch of Proof: verify that $\tau_i (G_{ER})$ obeys the constraints of the ERD definition (definition 2.2).

Definition 4.1 - *Mapping* T_{man}.

Let G_{ER} and (R, K, I) be an ERD and its relational translate, by mapping T_e, respectively. Every Δ-transformation, τ_i, over G_{ER} maps to a schema restructuring manipulation over (R, K, I), σ_i, as follows:

(i) every vertex *connection* maps to a relation-scheme *addition*, and every vertex *disconnection* maps to a relation-scheme *removal;*

(ii) the sets I_i and I_i' of inclusion dependencies are the translates of the sets of *added and removed* edges, for vertex connections, and *removed* and *added* edges, for vertex disconnections, respectively;

(iii) the keys are computed exactly as in mapping T_e.

Proposition 4.2. Let G_{ER} and (R, K, I) be an ERD and its relational translate, by mapping T_e, respectively; let Δ and T_{man} be the set of ERD-transformations and the mapping of Δ to a set of schema restructuring manipulations, respectively. Then: (i) $T_{man}(\Delta)$ is a set of incremental and reversible restructuring manipulations; and (ii) $\forall\, \tau_i \in \Delta : T_e(\tau_i(G_{ER})) \equiv T_{man}(\tau_i)(T_e(G_{ER}))$.

Sketch of Proof: straightforward, by following the definition of T_{man}, together with the definitions of the Δ-transformations.

Definition 4.2 - *Vertex-Completeness.*

A set of ERD-transformations Λ is said to be *vertex-complete* iff

(i) every Λ–transformation maps to an incremental and reversible restructuring manipulation; (ii) for every ERD G_{ER}, there is a sequence of Λ–transformations which maps the empty diagram (G_{ER}) into G_{ER} (the empty diagram); and (iii) every e-vertex/r-vertex connection/disconnection that maps to an incremental and reversible restructuring manipulation belongs to Λ.

The following subset of *basic* Δ-transformations is sufficient to perform any vertex connection and disconnection, possibly at the cost of dismantling and reconstructing the ERD: connect and disconnect e-vertices and r-vertices without ingoing edges. Note that these basic Δ-transformations are restrictions of the Δ_1 and Δ_2 vertex connections and disconnections. However, it is both cumbersome and unnecessarily complex, to express a vertex connection or disconnection by a sequence of basic Δ-transformations, therefore requirement (iii) of the vertex-completeness.

Proposition 4.3. The set of Δ-transformations is vertex-complete.

Sketch of Proof: It is enough to analyze the vertex disconnections, since every vertex connection is the reverse of a vertex disconnection. For vertex disconnections, it is enough to study the disconnections that are *not allowed,* and these refer only to entity-sets with non-empty sets of dependent entity-sets or involved in relationship-sets.

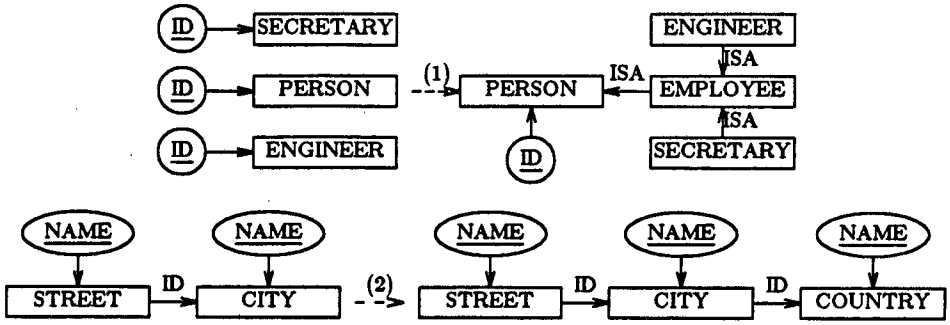


Fig.7 (1) *Connect* EMPLOYEE *isa* PERSON *gen* {SECRETARY,ENGINEER}
(2) *Connect* COUNTRY(NAME) *det* CITY

The role played by reversibility and incrementality in the specification of the Δ-transformations is illustrated by the two examples of figure 7: (i) the lack of reversibility does not allow to extend the connection of generic entity-sets to transformations such as that shown in figure 7(1); (ii) the lack of incrementality does not allow vertex connections such as that shown in figure 7(2).

V. RELATIONAL SCHEMA DESIGN

Traditional relational schema design consists mainly of a *normalization process* (cf. [13]). Relational normal forms have been developed in order to decrease both the impact of the side effects when changing relations, and the data redundancy in relations. The main cause of lack of normalization is the embedding into one relation of data about independent real-world facts. ER-consistent schemas favor the realization of many of the relational normalization objectives, because ER-oriented design simplifies and makes natural the task of keeping independent facts separated.

The design of relational schemas involving key and inclusion dependencies is a difficult task, mainly because of the excessive power of the inclusion dependencies. Consequently, Sciore [12] has proposed to constrain the set of inclusion dependencies to be *acyclic* and *key-based*. A design methodology pursuing the above properties is presented in [7]. However, the flatness of the pure relational environment of [7] defeats the declared intention of a simple and natural design. The acyclicity and key basing of the set of inclusion dependencies are captured in a precise manner by ER-consistency. Unlike traditional normalization procedures, the methodology proposed in [7] is *interactive*. The step-by-step schema development proposed by [7] can be straightforwardly achieved using the schema restructuring manipulations proposed in section 4. Assume, for instance, that the ERD of figure 8(i) is the result of a first design step, where EN, DN, and FLOOR, denote employee number, department number, and department floor, respectively. Next, it is decided that DEPARTMENT is, in fact, an independent entity-set, rather than an attribute of WORK, so that the ERD of figure 8(ii) results from the mapping specified by the following Δ_3-

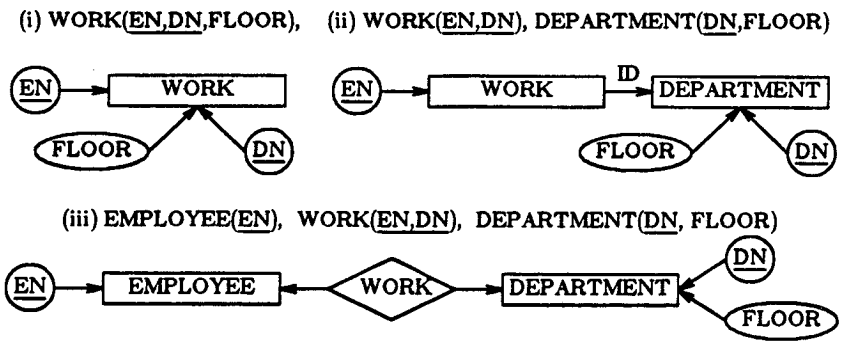


Fig.8 Entity-Relationship Consistent Relational Schemas.

transformation (conversion of an identifier-attribute into a weak entity-set): *Connect* DEPARTMENT(DN, FLOOR). A final step could be the disembedding of EMPLOYEE from WORK, as reflected by the ERD of figure 8(iii) which results from the mapping specified by the following Δ_3-transformation (conversion of a weak entity-set into an independent entity-set): *Connect* EMPLOYEE *con* WORK. All the examples presented in section 4 (figures 3-6) can be considered as additional examples of interactive design of ER-consistent schemas.

While [7] allows specifications that might imply unwanted properties, such as the cyclicity of the set of inclusion dependencies, and proposes correction transformations, our manipulations keep invariant the required schema characteristics, namely the key-basing and acyclicity of the set of inclusion dependencies. The incrementality and reversibility we have imposed on the set of restructuring manipulations insure a smooth schema evolution. The interactive design of ER-consistent relational schemas based on our restructuring manipulations has both top-down and bottom-up flavors. However, unlike the top-down or bottom-up relational schema normalizations, the schema development proposed by us reflects the evolution of a certain view of the information structure, rather than being based on the, more obscure, existence of dependencies.

When the design involves large information systems, it is split usually into the design of small *view schemas*, possibly for different user groups. Eventually the *view* schemas are *integrated* into a single global schema. The two main approaches to view integration are represented by [4] and [12] respectively. The view integration of [4] is done in a relational environment: a *combination* stage consists of defining inter-view dependencies, and an *optimization* stage consists of minimizing the redundancy in the global schema obtained in the combination stage. Besides the complexity problems of the optimization stage, the process does not preserve the assumed ER-consistency of the database. Unlike [4], [11] places the view integration in an explicit ER-oriented environment and proposes to accomplish it interactively. The integration process described in [11] is based mainly on the establishment of naming conventions (solving of synonyms and homonyms) and the specification of correspondences between entity/relationship-sets from different

views. The various integration options are thoroughly discussed and classified in [11], but *no operations* enabling a designer to align views for comparison and integration, and to actually perform the integration, are proposed. We claim that the restructuring manipulations defined by us fulfill the role of such operations. The incrementality and reversibility of the set of restructuring manipulations ensure a smooth view integration, and the completeness of the set ensures that all the possible integration options can be covered.

The space limit constrains us to give only two examples, adapted from [11], illustrating how view integration can be specified with our restructuring manipulations. Both examples refer to the ERDs of figure 9; since name similarities could be misleading, we suffix all vertex names by the corresponding view index. The first example refers to views (v1) and (v2), each consisting of a relationship-set ENROLL associating entity-sets COURSE and CS_STUDENT, respectively GR_STUDENT; it is known that the CS_STUDENT and GR_STUDENT entity-sets overlap, the two COURSE entity-sets are identical, and the two ENROLL relationship-sets are ER-compatible; accordingly, the integration of (v1) and (v2) resulting in global schema (g1) is specified by the following sequence of transformations:

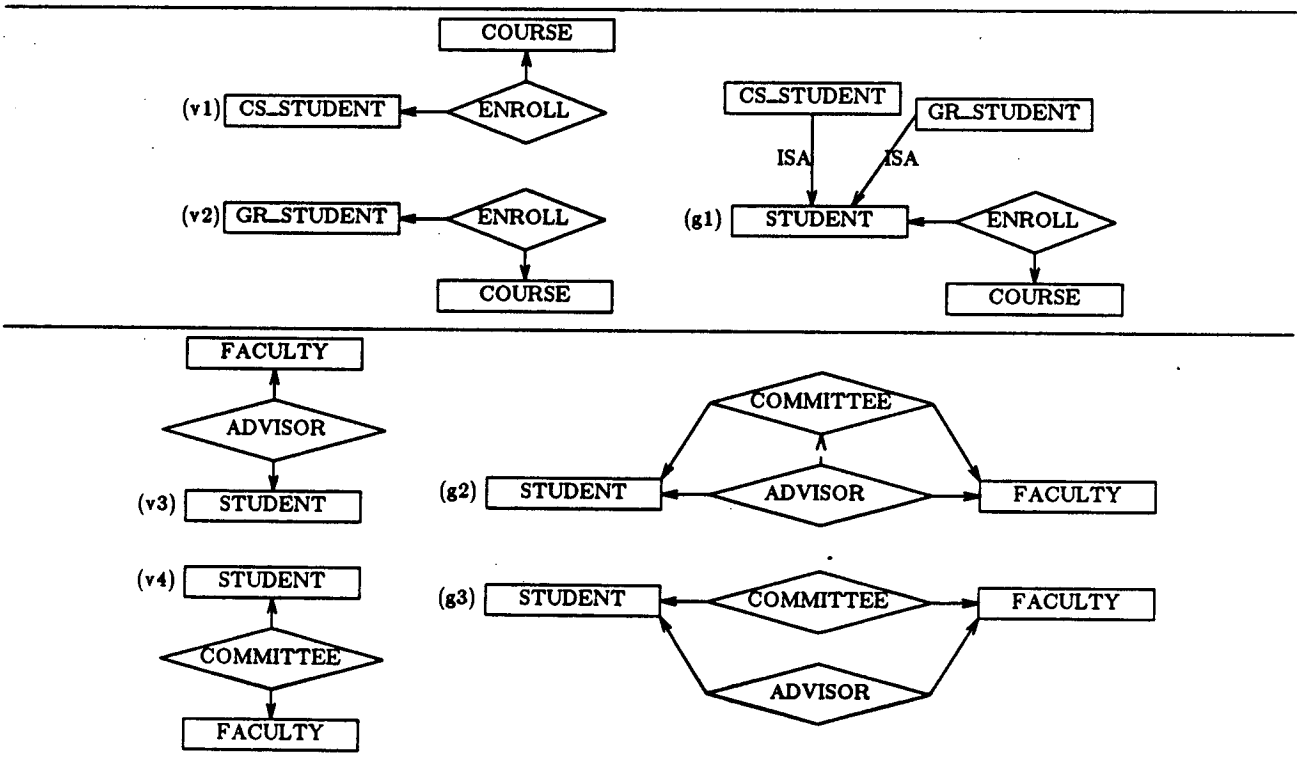


Fig.9 View Integration Examples.

(1) *Connect* STUDENT *gen* {CS_STUDENT,GR_STUDENT};
(2) *Connect* COURSE *gen* {COURSE_1,COURSE_2};
(3) *Connect* ENROLL *rel* {STUDENT,COURSE} *det* {ENROLL_1,ENROLL_2};
(4) *Disconnect* ENROLL_1; *Disconnect* ENROLL_2;
(5) *Disconnect* COURSE_1; *Disconnect* COURSE_2.

where (1) expresses the generalization of the overlapping entity-sets, (2) coupled with (5) express the generalization of the identical entity-sets, and (3) coupled with (4) express the merging of the two ER-compatible relationship-sets.

The second example refers to views (v3) and (v4), each consisting of two entity-sets, STUDENT and FACULTY, associated by relationship-sets ADVISOR and COMMITTEE, respectively; it is known that the two STUDENT entity-sets are identical, the two FACULTY entity-sets are identical, and the ADVISOR relationship-set is a subset of the COMMITTEE relationship-set; accordingly, the integration of (v3) and (v4) resulting in global schema (g2) is specified by the following sequence of transformations:
(1) *Connect* STUDENT *gen* {STUDENT_3,STUDENT_4};
(2) *Connect* FACULTY *gen* {FACULTY_3,FACULTY_4};
(3) *Connect* COMMITTEE *rel* {STUDENT,FACULTY} *det* COMMITTEE_4;
(4) *Connect* ADVISOR *rel* {STUDENT,FACULTY} *det* ADVISOR_3 *dep* COMMITTEE;
(5) *Disconnect* ADVISOR_3; *Disconnect* COMMITTEE_4;
(6) *Disconnect* STUDENT_3; *Disconnect* STUDENT_4;
(7) *Disconnect* FACULTY_3; *Disconnect* FACULTY_4.

where (1) and (2), coupled with (6) and (7), respectively, express generalizations of identical entity-sets, and (4) expresses the integration of relationship-set ADVISOR as a subset of COMMITTEE; the integration of (v3) and (v4) resulting in global schema (g3) is specified by the same sequence of restructurings as above, but with (4) replaced by: *Connect* ADVISOR *rel* {STUDENT,FACULTY} *det* ADVISOR_3 that expresses the integration of ADVISOR as an independent, rather than a subset, relationship-set.

VI. CONCLUSION

Schema restructuring is part of both database design and database reorganization, which are expressions of the specification and evolution of an information system. Since the capability of relational schemas to model information oriented systems is expressed by ER-consistency, we have investigated the restructuring of relational schemas in ER-consistent environments.

A natural continuation of our work would be to incorporate more semantic modeling capabilities into the high-level ER-oriented interface. Some of the possible extensions are mentioned below. (i) *Roles* express the functions played by entity-sets in relationship-sets. Roles are essential to distinguish the different involvements of an entity-set in a same relationship-set, and could relax constraint (ER3) of the ERD definition. The introduction of roles seems straightforward but tedious. (ii) *Multivalued attributes* are

directly supported by *one-level nested relations* [6], that is, relations with nesting done only over single basic attributes. Assuming that identifier attributes are not multivalued, the mappings between ERDs and relational schemas are unchanged, because key and inclusion dependencies involve only identifier attributes, and the sets of restructuring manipulations and reorganization operations have to undergo only minor changes. (iii) *Disjointness constraints* specify the disjointness of ER-compatible entity/relationship-sets. For instance, disjointness constraints can express the *partitioning* of a generic entity-set into disjoint specialization entity-subsets. Disjointness constraints are expressed in the relational model by *exclusion dependencies* [4].

Acknowledgement : The work of the first author was partially supported by the Applied Mathematical Sciences Research Program of the Office of Energy Research, U.S. Department of Energy, under contract number DE-AC03-76SF00098.

REFERENCES

[1] C. Batini and M. Lenzerini, "A computer-aided methodology for conceptual data-base design", *Information Systems*, **7**,3, 1982, pp. 265-280.

[2] C. Batini and M. Lenzerini, "Methodology for data schema integration in the entity-relationship model", *IEEE Trans. on Soft. Eng.* **SE-10**,6, Nov. 1984, pp. 650-664.

[3] S.S. Cosmadakis and P.C. Kanellakis, "Equational Theories and Database Constraints", *17th ACM Symposium on Theory of Computing*, 1985, pp. 273-284.

[4] M.A. Casanova and V.M.P. Vidal, "Towards a sound view integration methodology", *ACM Symposium on Principles of Database Systems* , 1983, pp. 36-47.

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

[6] P.C. Fisher and D. Van Gucht, "Determining when a structure is a nested relation", *11th VLDB Conference* , 1985, pp. 171-180.

[7] H. Mannila and K-J. Raiha, "Inclusion dependencies in database design", *2nd Conf. on Data Engineering* , 1986, pp. 713-718.

[8] 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, 1986, pp. 306-322.

[9] V.M. Markowitz, "Entity-relationship consistency for relational databases", Ph.D. thesis, Technion, June 1987.

[10] V.M. Markowitz and J.A. Makowsky, "Incremental reorganization of relational databases", *13th VLDB Conference* , 1987, pp. 127-135.

[11] S. Navathe, R. Elmasri, and J. Larson, "Integrating user views in database design", *IEEE Computer* **19**,1, Jan. 1986, pp. 50-62.

[12] E. Sciore, "Inclusion dependencies and the universal instance", *ACM Symposium on Principles of Database Systems* , 1983, pp.48-57.

[13] J.D. Ullman, *Principles of database systems* (second edition), Computer Science Press, 1982.

LAWRENCE BERKELEY LABORATORY
TECHNICAL INFORMATION DEPARTMENT
UNIVERSITY OF CALIFORNIA
BERKELEY, CALIFORNIA 94720