INCREMENTAL RESTRUCTURING OF RELATIONAL SCHEMAS

by

V.M. Markowitz and J.A. Makowsky

Technical Report #442

December 1986
INCREMENTAL RESTRUCTURING OF RELATIONAL SCHEMAS
Victor M. Markowitz and Johann A. Makowsky
Technion - Israel Institute of Technology

ABSTRACT

Schema restructuring is part of both database design and database reorganization, which are expressions of the specification and evolution of an information system. 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 Entity-Relationship diagram. We restrict our research of relational schema restructuring to ER-consistent schemas.

The basic relational schema restructuring manipulations are the addition and removal of relation schemes, together with the adjustment of inner and inter-relational dependencies. For relational schemas consisting of relation schemes, together with key and inclusion dependencies, we formally define the concept of incremental schema modification. Incrementality characterizes smooth schema transformations, without major disruptions.

We define a set of ER-diagram transformations representing various information manipulations, and define their mapping to relational schema restructuring manipulations. We prove the correctness of the mapping, and show that the resulting schema restructuring manipulations are incremental. While incrementality characterizes one-step schema modifications, reversibility assures that every such modification can be undone also in one step; the proposed ER-diagram transformations are shown to be reversible. We define two types of completeness for the ER-diagram transformations, ERD-completeness and vertex-completeness. A subset of basic ER-diagram transformations is shown to be ERD-complete, while the whole set of the ER-diagram transformations is shown to be vertex-complete.

Email[BITNET]: {victor, janos}@TECHSEL
I. INTRODUCTION.

The basic relational schema restructuring manipulations are the addition and removal of relation schemes, accompanied by the modification of the various dependencies. Schema restructuring is part of both database design and database reorganization, which are expressions of the specification and evolution of an information system.

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. User-system communication requires a database schema based on terms natural to the concepts employed by the users. Our choice for such a schema is the entity-relationship schema. Entity-Relationship (ER) oriented design [Chen] reflects a natural, although limited, view of the world: entities are qualified by their attributes and interactions between entities are expressed by relationships. ER-schemas are expressible in diagrammatic form called ER-diagram (ERD). In [MMR] we have investigated the significance of requiring from a relational database schema to comply with an entity-relationship structure, that is, to be representable by an 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.

We define a set of ER-diagram transformations covering both schema specification and restructuring, and representing various information manipulations. We complete the mappings between ER-diagrams and relational schemas with the mapping of ER-diagram transformations to restructuring manipulations over ER-consistent relational schemas. We prove the correctness of the mapping, and discuss the effect of the schema restructuring on the database state.

Relational schema restructuring has been perceived mainly as consisting of relation merging/splitting transformations, centered on relational algebra (e.g. [ST]). Besides being based on schemas without inclusion dependencies, these approaches overlook the information structure aspect of the schema manipulations, and do not pay attention to the topic of incremental schema evolution. Incremental relational schema design has been informally introduced in [MR] as characterizing smooth schema evolution, without major disruptions. The schema design methodology of [MR] refers only to the addition of relation schemes, and neglects the key interrelationships, which are essential in assuring smooth schema development. Moreover, the flatness of the pure relational environment of [MR] defeats any intention of a simple and natural design. We define formally the concept of incremental schema modification, and show that the ERD transformations map to incremental relational schema restructuring manipulations.
While incrementality characterizes one-step schema modifications, reversibility assures that every such modification can be undone also in one step; the proposed ERD transformations are shown to be reversible. We define two types of completeness for the ERD transformations, ERD-completeness, and vertex-completeness. Informally, ERD-completeness expresses the capability of constructing any ERD, while the vertex-completeness requires any vertex connection/disconnection to be an atomic transformation. A subset of basic ERD transformations is shown to be ERD-complete, while the whole set of the ERD transformations is shown to be vertex-complete.

The second section of the paper introduces ER diagrams. In section 3 we review the concept of ER-consistent relational schemas. In section 4 we define and exemplify a set of ERD transformations. Next, in section 5 we define the concept of incrementality for schema restructurings, and show that the ERD transformations map to incremental relational schema restructuring manipulations. Also in section 5 we discuss the effect of the restructuring manipulations on a non-empty database state. In section 6 we define the concept of reversibility for ERD transformations, and show that the ERD transformations proposed in section 4 are reversible. The ERD transformation completeness is defined and proved also in section 6. We close the paper by drawing some conclusions and outlining directions for further research.

II. THE ENTITY-RELATIONSHIP DIAGRAM.

Entity-Relationship oriented [Chen] design 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 perceived as the sharing of a same set of attributes. A value-set is a special kind of entity-set, without attributes, and grouping atomic values of a certain type. A relationship represents the interaction of several entities, and relationships of the same type are grouped in a relationship-set. A relationship-set can have attributes, just like an entity-set. An attribute is associated with one or several value-sets. A subset of the attributes associated with an entity-set may be specified as the, not necessarily unique, entity-identifier. ER-schemas are expressible in a diagrammatic form called ER-diagram (ERD). Entity-sets, relationship-sets, and attributes of entity-sets or relationship-sets, are represented by entity, relationship and attribute vertices, respectively. In the present paper we omit the specification of the value-sets for the sake of conciseness.
Definition 2.1 - Entity-Relationship Diagram (example in figure 1).

An Entity-Relationship Diagram (ERD) is a finite labeled digraph $G_{ER}=(V, H)$, where $V$ is the disjoint union of three subsets of vertices:

(i) e-vertices ($E$); (ii) r-vertices ($R$); and (iii) a-vertices ($A$);

e-vertices, a-vertices and r-vertices are represented graphically by rectangles, circles and diamonds, respectively;

$H$ is the set of directed edges, where an edge can be of one of the following forms:

(i) $E_i \rightarrow A_j$ ; (ii) $R_i \rightarrow A_j$ ; (iii) $E_i \rightarrow E_j$ ; and (iv) $R_i \rightarrow E_j$.

The reduced ERD of an ERD $G_{ER}$, is the subgraph of $G_{ER}$, $G'_{ER} = (V', H')$, where $V' = E \cup R$, and $H' = H - \{X_i \rightarrow A_j \in G_{ER}\}$.

Every vertex is labeled by the name of the associated entity-set, relationship-set, or attribute name; e-vertices, r-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.

Notations:

- $Atr(X_i) = \{A_j | X_i \rightarrow A_j \in G_{ER}\}$, denotes the set of a-vertices connected to an e-vertex/r-vertex $X_i$;
- $Id(E_i) \subseteq Atr(E_i)$, denotes the entity-identifier specified for e-vertex $E_i$;
- $Ent(E_i) = \{E_j | E_i \rightarrow E_j \in G_{ER}\}$, denotes the set of e-vertices to which an e-vertex $E_i$ is connected.
ERD edges specify existence constraints:

\( (E_i \rightarrow E_j) \) the ISA relationship expresses a subset relationship between two entity-sets, as part of general-
ization hierarchies: \( E_i \) is said to be a specialization of \( E_j \), and \( E_j \) is said to be a generalization of \( E_i \);

\( (E_i \rightarrow E_j) \) 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 (\( E_j \));

\( (R_i \rightarrow E_j) \) a relationship can exist only when the related entities also exist.

Definition 2.2 - Directed Paths.
A dipath in an ERD is called: ISA-path – if all the edges on the path are ISA-edges; ID-path – if all the vertices on the path are e-vertices, and at least one edge on the path is an ID-edge; and \( R \)-path – if the first vertex on the path is an r-vertex.

Definition 2.3 - Association Cardinality.
(Association) cardinality constraints are restrictions on maximum number of entities from a given entity-
set, that can be related, in the context of some relationship-set, to a specific combination of entities from other entity-sets. An edge \( R_i \rightarrow E_j \) is labeled by 1 if it corresponds to a cardinality of one; and by \( n \) if it corresponds to a cardinality of \( n \geq 1 \); we shall assume that at least one outgoing edge of every r-vertex has cardinality \( n \).

Definition 2.4 - Cluster.
Let \( G_{ER}=(V,E) \) be an ERD, and \( E_i \in G_{ER} \), an e-vertex such that there is no \( E_j : E_i \rightarrow E_j \in G_{ER} \); the cluster with root \( E_i \), \( Cluster(E_i) \), is defined as follows: \( Cluster(E_i) = E_i \cup \{ E_j | E_j \rightarrow E_i \in G_{ER} \} \).
In figure 1, for instance, \( Cluster(PERSON) = \{PERSON, EMPLOYEE\} \).

Definition 2.5 - Compatibility, Quasi-Compatibility.
The entity-set, relationship-set and attribute compatibility, whose intuition is straightforward, have the fol-
lowing graph-oriented analogs:

(i) two a-vertices, \( A_i \) and \( A_j \), are said to be compatible iff they are associated with the same value-sets;

(ii) two e-vertices, \( E_i \) and \( E_j \), are said to be compatible iff they belong to a same cluster, and quasi-
compatible if their respective identifiers are non-empty and compatible, and \( Ent(E_i) = Ent(E_j) \);
(iii) two r-vertices, \( R_i \) and \( R_j \), are said to be compatible iff there is a one-to-one correspondence, of compatible e-vertices between \( \text{Ent}(R_i) \) and \( \text{Ent}(R_j) \):

\[
\text{Comp}(R_i, R_j) = \{(E_k, E_m) | E_k \in \text{Ent}(R_i), E_m \in \text{Ent}(R_j), E_k \text{ and } E_m \text{ are compatible} \}.
\]

Definition 2.6 - Well-Formed ERD.

An ERD, \( G_{ER} \), is said to be well-formed iff it obeys the following constraints:

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

(ER2) \( \forall A_i \in G_{ER} : \text{indegree}(A_i) = 1; \)

(ER3) \( \forall E_i \in G_{ER} : \) if \( E_i \) has outgoing ISA-edges then: \( \text{Id}(E_i) = \emptyset \); all the e-vertices in \( \text{Ent}(E_i) \) are compatible; and \( E_i \) has no incident ID-edges;

otherw.ise: \( \text{Id}(E_i) \neq \emptyset \); 

(ER4) \( \forall R_i \in G_{ER} : \text{indegree}(R_i) = 0 \) and \( \text{Ent}(R_i) \geq 2; \)

(ER5) for any two ID-paths/R-paths starting in a same e-vertex/r-vertex, \( X_i : X_i \) is the unique vertex belonging to both these paths.

Directed cycles could consist only of ISA and ID edges. Constraint (ER1) above guarantees that such cycles do not exist; the meaning of this constraint is that 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 or relationship-set, therefore constraint (ER2). Constraint (ER5) is introduced in order to avoid the use of roles; it simplifies our presentation, by assuring, for instance, the uniqueness of the correspondence of two compatible relationship-sets.

III. ENTITY-RELATIONSHIP CONSISTENT RELATIONAL 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 \( R \). A relation scheme is a named set of attributes, \( R_i(A_i) \). On the semantic level, every attribute is assigned a domain. 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.

We deal with two kinds of dependencies, one inner relational, and one inter relational:
(i) a 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 $t[Y] = t'[Y]$; a key dependency over $R_i(A_i)$, is an FD $K_i \rightarrow A_i$, where $K_i \subseteq A_i$, and no subset of $K_i$ has this property; $K_i$ is called key;

(ii) an inclusion dependency (IND) is a statement of the form $R_i[X] \subseteq R_i[Y]$, 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 $r_i[X] \subseteq r_j[Y]$. 

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

**Definition 3.1 - Inclusion Dependency Graph.**

For a set of INDs, $I$, over $R$, the associated IND graph (IG) is the digraph $G_i=(V, E)$, where $V=R$ and $R_i \rightarrow R_j$ is an edge of $E$ iff $R_i[X] \subseteq R_j[Y]$ is in $I$.

**Definition 3.2 - Inclusion Dependency Properties.**

(i) An IND $R_i[X] \subseteq R_j[Y]$, is said to be key-based [Sci] if $Y = K_j$ and typed [CV] if $X = Y$.

(ii) A set of INDs, $I$, is said to be bounded [Sci] if whenever $R_i[X_i] \subseteq R_j[X_j]$, and $R_i[X_i] \subseteq R_k[X_k]$, there is $R_m$ s.t. $R_j[X_j] \subseteq R_m[X_m]$, and $R_k[X_k] \subseteq R_m[X_m]$, are implied by $I$.

(iii) A set of INDs, $I$, is said to be cyclic if either $R_i[X_i] \subseteq R_j[Y_j]$ for $X \neq Y$, or there are $R_1,...R_n$ such that $R_i[X_i] \subseteq R_1[Y_1], R_1[X_1] \subseteq R_2[Y_2],..., R_n[X_n] \subseteq R_i[Y_i]$; $I$ is acyclic iff the associated IG digraph is a dag [Sci].

**Definition 3.3 - Correlation Key.**

Given a relation scheme $R_i$, a correlation key (CK) in $R_i$ is a subset of $A_i$, appearing as a key in some relation different from $R_i$; the union of all correlation keys associated with $R_i$ is denoted $CK_i$.

**Definition 3.4 - Key Graph.**

For a set of keys, $K$, over $R$, the associated key graph (KG) is a digraph $G_k=(V, E)$, where $V=R$ and $R_i \rightarrow R_j$ is an edge of $E$ iff (i) $CK_i = K_i = K_j$; or (ii) $K_j \subseteq CK_i$ and there is no $R_k$ such that $K_j \subseteq CK_k$ and $K_k \subseteq CK_i$.

**Proposition 3.1 (Theorem 5.1 [CV]).**

Given a set of typed INDs, $I$, every IND $R_i[X] \subseteq R_j[Y]$ 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 IG digraph, corresponding to a sequence of INDs of $I$, $R_i[W] \subseteq \cdots \subseteq R_j[W]$, such that $X \subseteq W$. 

Technion - Computer Science Department - Technical Report CS0442 - 1986
Proposition 3.2 (Theorem 5.3 [CV]).
Let \( I \) and \( K \) be sets of INDs and keys, respectively in a relation schema \((R,K,I)\); then 
\[
(I \cup K)^+ = I^+ \cup K^+.
\]

In [MMR] we have presented the direct mapping (figure 2) and reverse mapping between ER-diagrams and relational schemas of the form \((R,K,I)\). We review below some results of [MMR] and extend the characterization of ER-consistent relational schemas.

Definition 3.5 - Entity-Relationship Consistency.
A relational schema \((R,K,I)\) is said to be ER-consistent either if it is the translate of an ERD, or if it can be mapped to an ERD. A relational database with an ER-consistent schema is said to be ER-consistent.

Proposition 3.3 (Lemma 4.1 [MMR]).
Let \((R,K,I)\) be the relational schema 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 digraphs associated with \((R,K,I)\), respectively.
\(G_I\) is a subgraph of \(G_K\), and \(G_I\) and \(G'_{ER}\) are isomorphic.

Proposition 3.4. (Proposition 4.3 [MMR]).
Let \((R,K,I)\) be the relational schema translate of an ERD \(G_{ER}: I\) is typed, key-based, acyclic and bounded.

Proposition 3.5
Let \((R,K,I)\) be an ER-translate relational schema; if \(R_i[K_j] \subseteq R_j[K_J]\) and \(R_j[K_m] \subseteq R_m[K_m]\) belong to \(I\), then \(K_m \subseteq K_j\).
Proof: straightforward from the specification of the ERD to relational schema mapping.

From propositions 3.1, 3.4 and 3.5 follows:

Corollary 3.6
Let \((R,K,I)\) be an ER-translate relational schema; an IND \(R_i[X] \subseteq R_j[Y]\) 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 IG digraph.

Notation: typing and key-basing allow to denote an IND \(R_i[K_j] \subseteq R_j[K_J]\) as \(R_i \subseteq R_j\).
\[ T_s : \text{Mapping ER-Diagram Into Relational Schema} \]

Input: \[ G_{ER} = (V, H), \text{ an ERD}; \]

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 \(E_i\) and r-vertex \(R_j\) compute the following sets of a-vertices:
   \[
   \text{Key}(E_i) = \text{XKey}(E_i) = \text{Id}(E_i) \cup \{ \text{Id}(E_j) | E_i \rightarrow E_j \in G_{ER}\};
   \]
   \[
   \text{Key}(R_j) = \bigcup \{ \text{Key}(E_j) | R_i \rightarrow E_j \in G_{ER}\}, \quad \text{XKey}(R_j) = \{ \text{Key}(E_j) | R_i \rightarrow E_j \in G_{ER}\};
   \]

3. for every e-vertex/r-vertex \(X_i\): define relation-scheme \(R_i\);
   \[
   K_i := \text{Key}(X_i); \quad A_i := \text{Attr}(X_i) \cup \text{XKey}(X_i);
   \]
   \[
   K := K \cup \{ K_i \}; \quad R := R \cup R_i(A_i);
   \]

4. let \(R_i\) and \(R_j\) be two relation schemes corresponding to e-vertex/r-vertex \(X_i\) and e-vertex \(E_j\), respectively; for every edge \(X_i \rightarrow E_j\): \(I := I \cup (R_i[K_i] \subseteq R_j[K_j])\).

relation-schemes (keys are underlined):

\[
\begin{align*}
\text{CITY(CITY.NAME)} \\
\text{STREET(STREET.NAME,CITY.NAME)} \\
\text{PERSON(PERSON.ID,NAME)} \\
\text{EMPLOYEE(PERSON.ID,SALARY)} \\
\text{LIVES(PERSON.ID,STREET.NAME,CITY.NAME)}
\end{align*}
\]

inclusion dependencies:

\[
\begin{align*}
\text{STREET(CITY.NAME)} \subseteq \text{CITY(CITY.NAME)} \\
\text{LIVES(STREET.NAME,CITY.NAME)} \subseteq \text{STREET(STREET.NAME,CITY.NAME)} \\
\text{LIVES(PERSON.ID)} \subseteq \text{PERSON(PERSON.ID)} \\
\text{EMPLOYEE(PERSON.ID)} \subseteq \text{PERSON(PERSON.ID)}
\end{align*}
\]

Fig.2 Mapping ER-Diagram Into Relational Schema Procedure and Example.
IV. ENTITY-RELATIONSHIP DIAGRAM TRANSFORMATIONS

Schema restructuring is part of both database design and database reorganization. The basic relational schema restructuring manipulations are the addition and removal of relation schemes, together with the adjustment of inner and inter-relational dependencies. However, adding and removing relations are just expressions of information structure specification and evolution, and as such must have information structure transformations counterparts. Consequently, we restrict our research to the schema restructuring manipulations in an ER-consistent environment.

Definition 4.1 - ER-Consistency Preserving Transformation.
An ERD transformation which maps a well-formed ERD into another well-formed ERD is said to be ER-consistency preserving.

Definition 4.2 - ER-Vertex Transformation.
An ERD transformation consisting of a vertex connection/disconnection is called an ER-vertex transformation.

We present in this section a set of ER-vertex transformations, \( \Delta \), which are ER-consistency preserving.


The connection/disconnection of an attribute-vertex is a simple transformation with obvious prerequisites and ERD mapping; it has the following form: \( \text{Connect} / \text{Disconnect} \ A_i \ to \ from \ X_j \) where \( A_i \) and \( X_j \) denote an a-vertex and e-vertex/r-vertex, respectively. We use the connection/disconnection of attribute-vertices only embedded in other transformations.

2. Connect / Disconnect Relationship-Vertex.

The connection of a relationship-vertex represents the specification of a new relationship-set associating some existing entity-sets, with certain cardinalities; it has the following form:

\[ \text{Connect} \ R_i \ rel \ ENT \]

where \( R_i \) denotes an r-vertex, \( ENT \) denotes a set of pairs \( E_j(C_j) \), with \( E_j \) denoting an e-vertex and \( C_j \) denoting a cardinality label.
Prerequisites: (i) \( R_i \notin G_{ER} \), \( \forall E_j(C_j) \in ENT \): \( E_j \in G_{ER} \);

(ii) \( |ENT| \geq 2 \), and there are no compatible e-vertices in \( ENT \) (because of (ER5));

\( G_{ER} \) mapping: \( \text{add} \ R_i, \{ R_i \rightarrow E_j | E_j(C_j) \in ENT \} \).

The disconnection of a relationship-vertex is a simple transformation with obvious prerequisites and ERD mapping; it has the following form: \( \text{Disconnect} \ R_i \) where \( R_i \) denotes an r-vertex.


The connection of an entity-vertex represents the specification of a new entity-set, possibly related to existing entity-sets. The relation to existing entity-sets defines whether it is an independent, weak, specialization or generic entity-set.

3.1 Connect Independent/Weak Entity-Vertex (example in figure 3).

An independent/weak entity-set has a non-empty identifier, and other entity-sets may depend for identification on the new entity-set; a weak entity-set depends for identification on other entity-sets. The connection of an entity-vertex representing an independent/weak entity-set has the following form:

\[
\text{Connect} \ E_i(Id_i) \ [ \text{id} \ ENT \ ] \ [ \text{det} \ DEP \ ]
\]

where \( ENT \) denotes a set of e-vertices, \( DEP \) denotes a set of pairs \( E_j(Id_j) \), \( E_i \) and \( E_j \) denote e-vertices, and \( Id_i \) and \( Id_j \) denote sets of identifier a-vertices; \( E_i \) depends on every entity-set of \( ENT \), and every entity-set of \( DEP \) depends on \( E_i \);

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

(ii) \( \forall E_j(Id_j) \in DEP : E_j \in G_{ER}, \ Id_j \subseteq Id(E_j) \) and \( ENT \subseteq \text{Ent}(E_j) \);

(iii) \( \forall E_j(Id_j), \forall E_k(Id_k) \in DEP : Id_j \) and \( Id_k \) are compatible, and there is a

compatibility correspondence between them and \( Id_i \) — this correspondence implicitly defines the value-set association of the a-vertices of \( Id_i \);

(iv) if \( ENT \neq \emptyset \) then \( \forall E_j, \forall E_k \in ENT : \{ E_m | E_j \rightarrow E_m, \ E_k \rightarrow E_m \in G_{ER} \} = \emptyset \)

(enforcement of (ER5)).
3.2 Connect Generic/Specialization Entity-Vertex (example in figure 4).

A new entity-set can be specified as the specialization of several compatible entity-sets and, possibly, as the generalization of some compatible entity-sets; such an entity-set has an empty identifier. The connection of an entity-vertex representing a generic/specialization entity-set has the following form:

\[ \text{Connect } E_i \text{ isa ENT gen GEN,} \]

where \( E_i \) denotes an e-vertex, \( ENT \) and \( GEN \) denote sets of e-vertices; \( E_i \) is a specialization of every entity-set of \( ENT \), and generalizes the entity-sets of \( GEN \).
Prerequisites: (i) $E_j \in G_{ER}$, $\forall E_j \in (\text{ENT} \cup \text{GEN})$: $E_j \in G_{ER}$;

(ii) $\text{ENT} \cup \text{GEN}$ is a set of compatible e-vertices;

$G_{ER}$ mapping:

\[
\begin{align*}
\text{add} & E_i, \{E_i \rightarrow E_j | E_i \in \text{ENT}\}, \text{ and } \{E_k \rightarrow E_i | E_k \in \text{GEN}\}; \\
\text{remove} & \{E_k \rightarrow E_j | E_k \in \text{GEN}, E_j \in \text{ENT}\};
\end{align*}
\]

3.3 Disconnect Entity-Vertex (examples in figures 3 and 4).

The disconnection of an entity-set causes the distribution of its identifier attributes among the entity-sets depending on it, and the disconnection of the relationship-sets involving it. The disconnection of an entity-set is prohibited either when it might split a compatibility cluster, or when it is involved in relationship-sets (then the relationship-sets must be first removed). The disconnection of an entity-vertex has the following form:

\textbf{Disconnect } E_i

where $E_i$ denotes an e-vertex;

Prerequisites: (i) $E_i \in G_{ER};$

(ii) if $\{E_j | E_i \rightarrow E_j \in G_{ER}\} = \emptyset$ then $|\{E_k | E_k \rightarrow E_i \in G_{ER}\}| = 1$ (avoids the splitting of a compatibility cluster);

(iii) $\exists R_j \in G_{ER}: E_i \in \text{Ent}(R_j)$ (avoids removing an e-vertex connected to r-vertices);
$G_{ER}$ mapping:

**distribute**  
\{ $\textit{Connect} \cdot A'_k$ to $E_j \mid A'_k$ is a duplicate of $A_k \in Id(E_i) , E_j \rightarrow E_i \in G_{ER}$ \};

**add**  
$\{ E_j \rightarrow E_k \mid E_j \rightarrow E_i , E_i \rightarrow E_k \in G_{ER} \}$;  
$\{ E_j \rightarrow E_k \mid E_j \rightarrow E_i , E_i \rightarrow E_k \in G_{ER} , \text{ and} \}$  
\[ 3E_m , E_m \neq E_i , \text{ s.t. } E_j \rightarrow E_m \text{ and } E_m \rightarrow E_k \in G_{ER} \};$

**remove**  
$\{ E_j \rightarrow E_i \mid E_j \rightarrow E_i \in G_{ER} \}$;  
$\{ E_i \rightarrow E_k \mid E_i \rightarrow E_k \in G_{ER} \}$;  
$\{ E_i \rightarrow E_k \mid E_i \rightarrow E_k \in G_{ER} , \text{ and } E_i \}.$

**Proposition 4.1**

The $ER$-vertex transformations of the set $\Delta$ are $ER$-consistency preserving.

Proof: straightforward, by verifying, for every transformation, the enforcement of the constraints (ER1) to (ERS) of definition 2.6.

Several remarks concerning the above definitions are in order:

(i) the connection of entity-vertices embed the connection of the identifier attribute-vertices;

(ii) although seemingly the addition of a vertex would imply only additions of edges, some connection definitions include edge removals; generally the edge removals are mandatory, as in 3.1 (e.g. $ID \text{ STREET} \rightarrow \text{COUNTRY}$ in figure 3(2)), but sometimes the removal refers to extraneous edges, as in 3.2 (e.g. $ISA \text{ ENGINEER} \rightarrow \text{PERSON}$ in figure 4(1));

(iii) the constraints (ER1) to (ERS) guarantee, in general, information manipulation counterparts for the transformations, therefore the importance of its preservation; for instance, not removing the $ID \text{ STREET} \rightarrow \text{COUNTRY}$ edge in figure 3(2) is both meaningless and disobeying constraint (ERS);
(iv) there are transformations which are ER-consistency preserving, have an information manipulation meaning, however have not been taken into consideration (e.g. figure 5) because they are not incremental (incrementality, defined in the next section, characterizes smooth database evolution).

V. INCREMENTAL RESTRUCTURING OF ER-CONSISTENT RELATIONAL SCHEMAS

The basic restructuring manipulations of relational schemas of the form \((R, K, I)\), are the addition and removal of relation schemes, together with the appropriate modification of the key and inclusion dependencies. 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.

**Definition 5.1 - Incremental 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 \(R_i\); the restructuring manipulation is said to be incremental iff

- **addition** \(R' := R \cup R_i, K' := K \cup K_i, \text{ and } (I' \cup K')^+ = (I' \cup K_i)^+ \);  
- **removal** \(R' := R - R_i, K' := K - K_i, \text{ and } (I' \cup K')^+ = ((I \cup K)^+ - I_i - K_i)^+ \).

Note that verifying incrementality for unrestricted relational schemas, might be very expensive because of the complexity of the implication problem for inclusion and functional dependencies (see [CK]).

An example of lack of incrementality is given in figure 5: it is evident that on the relational schema level there will be necessary, besides adding the key associated with the new relation scheme, to modify other several keys.

Now we turn to the restructuring of ER-consistent relational schemas. For the beginning we assume that the database state is empty. Given an ERD digraph its relational interpretation is given by \(T_s\). The mapping \(T_s\) is completed by the mapping of the ERD transformations to relational restructuring manipulations, \(T_{man} :\)
Definition 5.2 - Relation Scheme Addition/Removal.

Let $G_{ER}$ be an ERD, mapped to $G'_{ER}$ by the connection/disconnection of e-vertex/r-vertex $X_j$; let $(R, K, I)$ be the relational schema translate of $G_{ER}$, and $R_j$ be the relational translate of $X_j$. The mapping of the connection/disconnection of an e-vertex/r-vertex $X_j$ is defined as follows:

**relational schema** $(R, K, I)$ maps to $(R', K', I')$, such that

**addition**

$R' := R \cup R_j$, $K' := K \cup K_j$, and $I' := I \cup I_j - I_i'$, where

$I_i = \{R_j \subseteq R_i | R_j \in R \} \cup \{R_i \subseteq R_j | R_j \in R \}$, and

$I_i' = \{R_j \subseteq R_k | R_j \subseteq R_k \in I \}$.

**removal**

$R' := R - R_j$, $K' := K - K_j$, and $I' := I - I_i \cup I_i'$, where

$I_i = \{R_j \subseteq R_i | R_j \subseteq R_i \in I \} \cup \{R_i \subseteq R_j | R_i \subseteq R_j \in I \}$, and

$I_i' = \{R_j \subseteq R_k | R_j \subseteq R_k \in I \}$.

Proposition 5.1

The relational schema restructuring of definition 5.2, is such that $(R', K', I')$ is the relational translate of $G'_{ER}$.

Proof: straightforward, following the definitions of vertex connection/disconnection.

Proposition 5.2

The relational schema restructuring of definition 5.2 is incremental.

Proof: based on proposition 3.2 and corollary 3.6.
When the database state associated with the relational schema is empty, the restructuring has no effect on the state. This is the case with the schema design. Database reorganization, on the other hand, involves non-empty states. Before discussing the propagation of schema restructuring on a non-empty database state, we need two ERD-based definitions. Both definitions have direct correspondents over the inclusion dependency digraph, however we prefer the ERD environment because of its clearer intuition.

Definition 5.3 - 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 $ENT = \{E_j | E_j \in G_{ER}\}$, denoted $\text{uplink}(ENT)$, iff $\forall E_j \in ENT : E_j \rightarrow E_i \in G_{ER}$ (possibly of length 0), and there is no other e-vertex with this property.

In figure 6, for instance, PERSON is the uplink(STAFF,STUDENT).

In an ER-consistent relational database, every relation corresponds to an entity-set or relationship-set, and every tuple represents an entity or relationship, respectively. A tuple insertion in an ER-consistent database refers to a relation associated with the relational translate of an e-vertex/r-vertex. Such an insertion embodies the enforcement of the existence constraints specified by the ERD edges, thus propagating over an ERD subgraph induced by the respective vertex.

Definition 5.4 - Insert Frame (example in figure 6).
Let $G_{ER}$ be an ERD, $G'_{ER}$ its associated reduced ERD, and $X_i$ an e-vertex/r-vertex of $G_{ER}$; the insert frame of $X_i$ is the following (acyclic) subgraph of $G_{ER}$:

$$G_{ER}(X_i^{\text{insert}}) = (V_i, H_i) : V_i = X_i \cup \{X_j | X_i \rightarrow X_j \in G'_{ER}\}, H_i = \{X_k \rightarrow X_j | X_k, X_j \in V_i, X_k \rightarrow X_j \in G'_{ER}\}.$$
Effect of Schema Restructuring on the Database State.

Let \( G_{ER} \) be an ERD, \( X_i \) an e-vertex/r-vertex of \( G_{ER} \), \( (R, K, I) \) the relational schema translate of \( G_{ER} \), and \( R_i \) the relational translate of \( X_i \). The effect of the connection/disconnection of an e-vertex/r-vertex \( X_i \) on the database state associated with \( (R, K, I) \), is as follows:

- **removal**: delete the relation associated with \( R_i \);
- **addition**: (a) set up an empty relation \( r_i \), and associate it with \( R_i \);
  
  \[ r_i := r_i \cup \bigcup_{R_j \subseteq R_i} r_j[K_j]. \]

While the removal evidently maps the database state to a consistent new state, the addition triggers an insert propagation from \( R_i \) to all the relations associated with the translates of the vertices of \( G_{ER}(X_{inty}) \). The following lemma gives the sufficient condition under which no propagation takes place.

**Lemma 5.3**

Let \( (R, K, I) \) be the relational schema translate of \( G_{ER} \), and \( R_i \) be the relational translate of connected/disconnected e-vertex/r-vertex \( X_i \). The addition of \( R_i \) maps the database state to a consistent state if \( \text{Ent}(X_i) \subseteq \{ \text{uplink } (X_j) \mid X_j \rightarrow X_i \in G_{ER} \} \).

Several remarks concerning the above propagation are in order:

(i) when \( X_i \) is an r-vertex, the relation associated with \( R_i \) is empty, and no propagation takes place;

(ii) when \( X_i \) is an independent/weak e-vertex, the condition of lemma 5.3 is met by prerequisite (ii) of definition 3.1 (section 4), so that no propagation takes place;

(iii) when \( X_i \) is a generic/specialization e-vertex, there is, generally, a propagation, but it takes place within one cluster, because of prerequisite (ii) of definition 3.2 (section 4).

Schema restructuring incrementality guarantees that only the immediate neighborhood of the added/removed relation scheme is affected. The analogous state modification incrementality, defined below, is a stronger requirement.
**Definition 5.5 - Incremental State Modification.**

A state modification is said to be **incremental** iff it consists of the addition or removal of a single relation.

Evidently, the lack of insert-propagation guarantees the incrementality of the state modification implied by a restructuring manipulation. Consequently, only the transformation defined in 3.2 (section 4) might imply a non incremental state modification; in order to avoid this, the definition can be restricted following lemma 5.3.

**VI. ER-VERTEX TRANSFORMATION COMPLETENESS**

The ER-vertex transformations presented in section 4 have been shown in section 5 to be incremental. In the present section we shall define the concept of reversibility and investigate the completeness of the set of ER-vertex transformations. First we extend the set \( \Delta \) of the ER-vertex transformations presented in section 4 with an additional entity-vertex connection which extends the generalization to incompatible entity-sets; this extended set of ER-vertex transformations will be denoted \( \Delta' \).

*Connect Generic Entity-Vertex* (example in figure 7).

A new entity-vertex with a non empty identifier can be defined as the generalization of several quasi-compatible entity-sets (i.e. entity-sets with compatible identifiers, and depending on the same set of entity-sets). The connection of an entity-vertex representing such a generic entity-set has the following form:

\[
\text{Connect } E_j(\text{Id}_j) \xrightarrow{\text{gen}} \text{GEN}
\]

where \( E_j \) denotes an e-vertex, \( \text{Id}_j \) denotes a set of a-vertices, and \( \text{GEN} \) denotes a set of e-vertices;

**Prerequisites:**

(i) \( E_i \not\in G_{ER} \) and \( \forall E_j \in \text{GEN} : E_j \in G_{ER} \);

(ii) \( \forall E_k, E_j \in \text{GEN} : \)

\[
\text{Ent}(E_k) = \text{Ent}(E_j), \text{ and let ENT denote this common set of e-vertices;}
\]

\[
\text{Id}(E_k) \text{ and } \text{Id}(E_j) \text{ are not empty and compatible, and there is a compatibility correspondence between them and } \text{Id}_i \text{ – this correspondence defines the value-set association of the a-vertices of } \text{Id}_i ;
\]
The relational schema restructuring consists of the addition-manipulation defined in definition 5.2, preceded by the renaming of all the corresponding compatible identifier attributes, to the name of the respective attribute of the new generic entity-set; consequently, the transformation is incremental.

**Definition 6.1 - Reversibility.**

Let \( \tau_i \) be an ERD transformation; \( \tau_i \) is said to be reversible iff there is another ERD transformation \( \tau_j \) such that for any ERD, the sequence of \( \tau_i \) and \( \tau_j \) applied on the ERD, returns the same ERD, up to a renaming of compatible a-vertices.

Reversibility has been exemplified in section 4, where every connection example is paired with a disconnection counterpart. While incrementality characterizes one-step schema modifications, reversibility assures that every such modification can be undone in one step. An example of an ERD transformation which is incremental and ER-consistency preserving, but not reversible, is given in figure 8.
Definition 6.2 - Basic ER-Vertex Transformations.
The following ER-vertex transformations are called basic: (i) connect and disconnect attribute-vertex; (ii) connect and disconnect relationship-vertex; (iii) connect and disconnect entity-vertex without ingoing edges.

Note that (iii) above is a restriction of the entity-vertex connection/disconnection defined in section 4/subsection 3, and that the basic ER-vertex transformations are evidently independent.

Definition 6.3 - ERD Transformation ERD-Completeness.
A set of ERD transformations is said to be ERD-complete iff every component transformation is incremental and reversible, and for every ERD $G_{ER}$, there is a sequence of transformations, which maps the empty diagram ($G_{ER}$) into $G_{ER}$ (the empty diagram).

Proposition 6.1
The basic ER-vertex transformations form an ERD-complete set.
Proof: straightforward.

The proposition 6.1 shows that the basic ER-vertex transformations are sufficient to perform any vertex connection/disconnection. However, it is both cumbersome and unnecessarily complex, to express a vertex connection/disconnection by a sequence of basic ER-vertex transformations. Consequently, a vertex connection/disconnection should be defined as an ER-consistency preserving, incremental and reversible atomic transformation.

---

Fig. 8 (1) Connect EMPLOYEE isa PERSON gen SECRETARY, ENGINEER
(2) Disconnect EMPLOYEE = (2) in figure 4.
Definition 6.4 - ER-Vertex Transformation Vertex-Completeness.
A set of ER-vertex transformations is said to be vertex-complete iff every ER-consistency preserving, incremental and reversible vertex connection/disconnection, is expressible by a single transformation of the set.

Proposition 6.2
The set $\Delta'$ of ER-vertex transformations is vertex-complete.
Sketch of the proof: the connection/disconnection of a-vertices and r-vertices are basic transformations; the disconnection of an e-vertex can be shown to cover all legal e-vertex disconnections, and then every different case of disconnection is shown to have as inverse one of the e-vertex connection forms.

Note: the state reversibility is analogous to the schema restructuring reversibility; it is evident that while vertex connections are state reversible, vertex removals are, generally, not state reversible.

VII. CONCLUSION
Schema restructuring is part of 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 relational schemas in an ER-consistent environment. We have defined two concepts characterizing smooth schema modification: incrementality, and reversibility. While incrementality characterizes the locality of one-step schema modifications, reversibility assures that every such modification can be undone also in one step.

We have proposed a set of ER-vertex transformations and defined their mapping to ER-consistent relational schema restructuring manipulations. We have proved the correctness of the mapping, and have shown that the transformations are reversible and map to incremental schema restructuring manipulations. Two types of completeness, ERD-completeness and vertex-completeness, have been proposed for the ERD transformations; a subset of basic ER-vertex transformations has been shown to be ERD-complete, while the whole set of the ER-vertex transformations has been shown to be vertex-complete.

The set of ER-vertex transformations does not encompass all possible ERD transformation. Rather than proposing more transformations, we have preferred to present a methodology of defining and analyzing them. Once the methodology is understood it is straightforward, although possibly tedious, to extend the
transformations for less restricted ERDs, or to add new transformations. We shall exemplify the later with two attribute-oriented transformations related to generalization: attribute-vertex unification and distribution.

*Attribute-vertex unification* and *distribution* refer to the redistribution of attributes within a generalization cluster: compatible attributes of entity-sets that are specializations of a same generic set, can be *unified* into an attribute of the generic entity-set, while an attribute of a generic entity-set can be *distributed* between some/all of its specialization entity-sets. Self-explaining examples of attribute unification and distribution are given in figure 9. For instance, the generalization of figure 7 can be followed by the unification of figure 9, and the disconnection of figure 7 can be preceded by the distribution of figure 9. The precise definitions and relational schema restructuring of these transformations are straightforward; they are trivially incremental and obviously reversible.
REFERENCES.


