INCREMENTAL REORGANIZATION OF RELATIONAL DATABASES

by

V.M. Markowitz and J.A. Makowsky

Technical Report #448
February 1987
INCREMENTAL REORGANIZATION OF RELATIONAL DATABASES
Victor M. Markowitz and Johann A. Makowsky
Technion - Israel Institute of Technology

ABSTRACT

The evolution of an information system is reflected in data modeling by database reorganization. Database reorganization consists of schema restructuring coupled with entity-conservative state modification. Since algebraic operations consist of the embedding of schema restructuring and state modification, relational database reorganization has been mostly centered on relational algebra. This approach fails to capture the information structure aspect of the database reorganization, mainly because the traditional relational model does not provide a suitable framework to deal with information. 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. For ER-consistent databases both the schema restructuring and the state modification are more complex than for traditional relational databases. The basic relational schema restructuring manipulations are the addition and removal of relation schemes, accompanied by the modification of the inner and inter-relation dependencies. Recently we have defined a set of incremental and reversible schema restructuring manipulations as the translates of a set of vertex-oriented ER-diagram transformations.

In the present paper we deal with the second component of database reorganizations, namely entity-conservative state modifications that keep invariant the set of entities in the database, and the reorganization operations as a whole. We introduce two operations that perform incremental, that is, propagation-free and single-relation, conservative state modifications. We prove that these operations form a complete, in a relational-calculus based sense, set. Database reorganization operations are defined as consisting of compatible pairs of incremental restructuring and conservative state modification manipulations. The proposed set of database reorganization operations is shown to be complete.

Authors' address: Computer Science Department, Technion, Haifa 32000, Israel; Email [BITNET]: {victor,jaanek}@TECHSEL
I. INTRODUCTION.

The evolution of an information system is reflected in data modeling by database reorganization [TL]. Database reorganization consists of schema restructuring accompanied by some entity-conservative state modification. Entity-conservative state modifications are state modifications that keep invariant the set of entities in the database. Since algebraic operations consist of the embedding of schema restructuring and state modification, relational database reorganization has been mostly centered on relational algebra (e.g., [ST]). This approach overlooks the information structure aspect of the database reorganization, mainly because 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.

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-diagrams (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.

For ER-consistent databases both the schema restructuring and the state modification are significantly more complex than for regular relational databases. The basic relational schema restructuring manipulations are the addition and removal of relation schemes, accompanied by the modification of the various dependencies. In [MM] we have defined a set of incremental and reversible schema restructuring manipulations as the translates of a set of vertex-oriented ER-diagram transformations. While incrementality characterizes the locality of one-step restructuring manipulations, reversibility assures that every such manipulation can be undone also in one step.

We introduce two operations that perform incremental, that is, propagation-free and single-relation, and conservative state modifications. We propose for such operations, over ER-consistent databases, a definition of completeness based on the relational calculus (Codd-completeness in [CH]). Relational calculus expressions over ER-consistent databases have to be compatible with the correspondent ER-consistent schema, just as for traditional relational databases. Consequently, only a notational adaptation is sufficient to derive from the relational calculus an ER-oriented version.
An Entity-Relationship Calculus (ERC) has been proposed in [AC]. Although inspired by the relational calculus, it is not clear how ERC relates to relational calculus, that is, what is the power of ERC. Moreover, while explicitly discarding the comparison of unrelated entity/relationship-sets, [AC] is not concerned whether ERC-expressions imply well-defined ER-structures. Subsequent to [AC] several ER-algebras have been proposed for the ER model ([MR], [PS], [CCE]). All these proposals fail to give a convincing answer to the question of what is the result of an ER-algebra operation. For instance in [MR] and [CCE] it is argued that any operation results in a new relationship-set, while in [PS], on the contrary, only entity-sets are created by algebraic operations. The main drawback of all these approaches is their ignoring of the role played by subsets; and of the dynamics of set interrelationships, both essential to algebraic manipulations.

Database reorganization operations are defined as consisting of compatible pairs of incremental restructuring and conservative state modification manipulations. The proposed set of database reorganization operations is shown to be complete.

The paper is organized as follows. The next section introduces ER diagrams. In section 3 we review the concept of ER-consistent relational databases. In section 4 we briefly review the tuple oriented version of relational calculus, and adapt it for ER-consistent databases. In section 5 we investigate state modifications in ER-consistent databases. Next, in section 6, we introduce two operations performing incremental state modifications, and prove their completeness. In section 7 we briefly review relational schema restructuring manipulations. In section 8 we define database reorganization operations as consisting of pairs of compatible schema restructuring and state modification manipulations, and prove their completeness. We close the paper by drawing some conclusions and outlining directions for further research.

II. THE ENTITY-RELATIONSHIP DIAGRAM.

Entity-Relationship 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. 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) $E_i \rightarrow E_j$; (iv) $R_i \rightarrow R_j$; and (v) $R_i \rightarrow E_j$.

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

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

---

**Fig.1 Entity-Relationship Diagram and Reduced Diagram Example (underlined identifiers).**
Notations:
- \( \text{Air}(X_i) = \{ A_j | X_i \rightarrow A_j \in G_{EX} \} \), denotes the set of e-vertices connected to an e-vertex/r-vertex \( X_i \);
- \( \text{Id}(E_i) \subseteq \text{Air}(E_i) \), denotes the entity-identifier specified for e-vertex \( E_i \);
- \( \text{Ent}(E_i) = \{ E_j | E_i \rightarrow E_j \in G_{EX} \} \), 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; \( E_i \) 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;
- \( (R_i \rightarrow R_j) \) a relationship can depend on the existence of related relationships.

Definition 2.2 - Directed Paths.
A digraph 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 and all the other vertices on the path are e-vertices.

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 \( G_{EX}=(V,H) \) be an ERD, and \( E_i \in G_{EX} \), an e-vertex such that \( \exists E_j : E_i \rightarrow E_j \in G_{EX} \); the cluster with root \( E_i \), Cluster \((E_i)\), is defined as follows: \( \text{Cluster}(E_i) = E_i \cup \{ E_j | E_i \rightarrow E_j \in G_{EX} \} \).
In figure 1, for instance, cluster (PERSON) is \{PERSON, EMPLOYEE, ENGINEER\}. 
Definition 2.5 - ER-Compatibility.
The entity-set, relationship-set and attribute compatibility, whose intuition is straightforward, have the following graph-oriented analogs:

(i) two a-vertices, \( A_i \) and \( A_j \), are said to be compatible iff they have the same type;
(ii) two e-vertices, \( E_i \) and \( E_j \), are said to be compatible iff they belong to a same cluster;
(iii) two r-vertices, \( R_i \) and \( R_j \), are said to be compatible iff there is a one-to-one correspondence, \( \text{Comp}(R_i, R_j) \), of compatible e-vertices between \( \text{Ent}(R_i) \) and \( \text{Ent}(R_j) \):

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

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;
otherwise: \( \text{Id}(E_i) \neq \emptyset \);
(ER4) \( \forall R_i \in G_{ER} : \text{Ent}(R_i) \geq 2 \) and
\[
\forall R_i \rightarrow R_j \in G_{ER} : \text{there is a one-to-one correspondence of e-vertices between } \\
\text{Ent}(R_i) \text{ and } \text{Ent}(R_j) : \\
\{(E_k, E_m) | R_i \rightarrow E_k, R_j \rightarrow E_m \in G_{ER}, \text{ such that } k_1 \leq k_2, \text{ and } \\
E_k \rightarrow E_m \in G_{ER} \text{ or } E_k \neq E_m \};
\]
(ER5) for any two ID-paths/R-paths starting in a same e-vertex/r-vertex, \( X_i \rightarrow X_j \) 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 (e.g. we do not allow a relationship-set to involve a same entity-set with different roles); it simplifies our presentation, by assuring, for instance, the uniqueness of the correspondence of two related relationship-sets (ER4).
III. ENTITY-RELATIONSHIP CONSISTENT RELATIONAL DATABASES.

A relational schema is a pair \( (R, D) \) where \( R \) is a set of relation schemes, \( R = (R_1, \ldots, R_k) \), 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 \( r = <D_1, \ldots, D_m, r_1, \ldots, r_k> \), where \( r_i \) is assigned a subset of the cartesian product of the domains corresponding to its attributes. Note: we shall assume that every element of any domain appears in at least one relation. 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 relationial, 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 \( r_i \) and \( t \), \( t[X] = r[X] \) implies \( t[Y] = r[Y] \); 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 \( R_i[X] \subseteq R_j[Y] \), where \( X \) and \( Y \) are subsets of \( A_i \) and \( A_j \), respectively, and \( |X| \leq |Y| \); an IND \( R_i[X] \subseteq R_j[Y] \) is valid in a state \( r \), iff \( r_i[X] \sqsubseteq r_j[Y] \).

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

Definition 3.1 - Inclusion Dependency Graph, Properties.

(i) For a set of INDs, \( I \), 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 \( R_i[X] \subseteq R_j[Y] \in I \).

(ii) An IND \( R_i[X] \subseteq R_j[Y] \) is said to be typed (CV) iff \( X = Y \).

(iii) A set of INDs, \( I \), 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 \( R_i[X_i] \subseteq R_j[Y_j] \), \( R_1[X_1] \subseteq R_2[Y_2] \), \ldots, \( R_n[X_n] \subseteq R_i[Y_i] \); \( I \) is acyclic iff the associated IND graph is a dag [Sci].

Definition 3.2 - Correlation Key, Extended Key, Key Graph.

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 \( CK_i \). The union of \( CK_i \) and \( K_i \) is denoted \( EK_i \), and is called the extended key of \( R_i \).
Mapping ER-Diagram Into Relational Schema

Input: \( G_{ER} = (V, H) \), an ERD (\( G'_{ER} \) is the reduced 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 e-vertex;

(2) for every e-vertex \( E_i \) and r-vertex \( R_j \) compute the following sets of a-vertices:

\[
\begin{align*}
Key(E_i) &= XKey(E_i) = Id(E_i) \cup \{Id(E_j) \mid E_i \rightarrow E_j \in G_{ER}\}; \\
Key(R_j) &= \bigcup \{Key(E_j) \mid R_i \rightarrow E_j \in G_{ER}\}. \quad XKey(R_j) = \{Key(E_j) \mid R_i \rightarrow E_j \in G_{ER}\};
\end{align*}
\]

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

\[
\begin{align*}
K_i := \text{Key}(X_i); \\
A_i := \text{Attr}(X_i) \cup XKey(X_i); \\
K := K \cup \{K_i\}; \quad R := R \cup R_i(A_i);
\end{align*}
\]

(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 \( X_i \rightarrow X_j \in G'_{ER} \):

\[
I := I \cup (\text{R}_i(K_j) \subseteq \text{R}_j(K_j)).
\]

relation-schemes (keys are underlined):

\[
\begin{align*}
\text{PERSON}(\text{PERSON.ID,N\text{AME}}) \\
\text{EM\text{PLOYEE}}(\text{PERSON.ID,\text{SA\text{LARY}}}) \\
\text{ENGINEER}(\text{PERSON.ID,\text{POSITION}}) \\
\text{DEPARTMENT}(\text{DEPARTMENT.DN}) \\
\text{\text{PROJECT}}(\text{PROJECT.PN}) \\
\text{WORK}(\text{PERSON.ID,DEPARTMENT.DN}) \\
\text{ASSIGN}(\text{PERSON.ID,PROJECT.PN,DEPARTMENT.DN})
\end{align*}
\]

inclusion dependencies:

\[
\begin{align*}
\text{\text{\text{EM\text{PLOYEE}}}}(\text{PERSON.ID}) \subseteq \text{PERSON}(\text{PERSON.ID}) \\
\text{ENGINEER}(\text{PERSON.ID}) \subseteq \text{\text{EM\text{PLOYEE}}}(\text{PERSON.ID}) \\
\text{WORK}(\text{PERSON.ID}) \subseteq \text{\text{EM\text{PLOYEE}}}(\text{PERSON.ID}) \\
\text{\text{\text{WORK}}}(\text{DEPARTMENT.DN}) \subseteq \text{DEPARTMENT}(\text{DEPARTMENT.DN}) \\
\text{ASSIGN}(\text{PERSON.ID}) \subseteq \text{ENGINEER}(\text{PERSON.ID}) \\
\text{\text{ASSIGN}}(\text{PROJECT.PN}) \subseteq \text{\text{PROJECT}}(\text{PROJECT.PN}) \\
\text{\text{ASSIGN}}(\text{PERSON.ID,DEPARTMENT.DN}) \subseteq \text{\text{WORK}}(\text{PERSON.ID,DEPARTMENT.DN})
\end{align*}
\]

Fig. 2 Mapping ER-Diagram Into Relational Schema Procedure and Example.
For a set of keys, $K$, over $R$, the associated key graph 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) $K_j \subseteq CK_i$ and there is no $R_k$ such that $K_j \subseteq CK_k$ and $K_k \subseteq CK_i$.

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 briefly review below some results of [MMR].

**Definition 3.3 - 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 whose schema is $ER$-consistent, is said to be an $ER$-consistent relational database.

**Proposition 3.1** (Lemma 4.1 and Proposition 4.3 [MMR]).
Let $(R, K, I)$ be the relational schema translate of the ERD $G_{ER}$, whose reduced ERD is $G'_{ER}$, and let $G_j$ and $G_k$ be the inclusion dependency and key digraphs associated with $(R, K, I)$, respectively. $J$ is typed and acyclic; and $G_j$ is a subgraph of $G_K$, and $G_j$ and $G'_{ER}$ are isomorphic.

**Proposition 3.2.**
Let $(R, K, I)$ be an ER-translate relational schema; an IND $R_j[X] \subseteq R_j[Y]$ 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.

Proof: consequence of theorem 5.1[CV], theorem 5.3[CV] and proposition 3.5[MM].

**Notation:** typing allows to denote an IND $R_j[EK_j] \subseteq R_j[EK_j]$, as $R_i \subseteq R_j$.

**IV. RELATIONAL CALCULUS IN $ER$-CONSISTENT DATABASES.**

We shall briefly review the tuple-oriented version of the relational calculus following [Pir], and then we shall adapt it for $ER$-consistent databases.

**Definition 4.1 - Tuple Relational Calculus (TRC).**
The syntax of TRC is defined as follows:

- **Terms**: constants, tuple variables, or indexed tuples; an indexed tuple is of the form $t[A]$, where $t$ is a tuple variable, and $A$ is an attribute;

- **Predicates**: unary range predicates, associated with a relation, and having as arguments tuple variables, or binary comparison predicates, whose arguments are constants and indexed tuples, such that the involved attributes are compatible;
propositions either predicates, or of the form \( P_1 \land P_2 \), \( P_1 \lor P_2 \), \(-P_1 \), \( P_1 \rightarrow P_2 \), where \( P_1 \) and \( P_2 \) are propositions;

quantiﬁers are range coupled, that is, of the form \((\exists x \in R)\), and \((\forall x \in R)\), where \( R \) is a range predicate;

formulas are propositions, or quantiﬁed formulas of the form \((\exists x \in R)\Phi(x)\), and \((\forall x \in R)\Phi(x)\), meaning \((\exists x)\langle R(x) \land \Phi(x) \rangle\) and \((\forall x)\langle R(x) \rightarrow \Phi(x) \rangle\), respectively, where \( R \) is a range predicate involving \( x \), \( x \) is free in \( \Phi \), \( \Phi \) does not contain range predicates for \( x \), and involves free variables other than \( x \); expression \( \{x_1 \cdots x_k \mid R_1(t_1) \land \cdots \land R_n(t_n) \land \Phi(t_1, \cdots, t_n)\} \), where \( x_i \) are indexed tuples, referring to \( n \) different tuple variables, \( t_1 \cdots t_n \), \( R_i \) are range predicates, \( \Phi \) is either absent or it is a formula with range-coupled quantifiers, without range predicates for \( t_i \cdots t_n \), and with \( t_1 \cdots t_n \) its only free variables.

An TRC-expression, \( \Psi \), can be perceived as associated with a virtual relation whose set of attributes is \( A_{\Psi} \triangleq \{A_i \mid t_i[A_i] \} \) appear in the headers of \( \Psi \), \( R_i \) is the range of \( t_i \), and \( A_i \in A_i \}, \) where every multiple appearance of a same attribute is renamed in order to assure the uniqueness of the attribute names. For ER-consistent databases we shall, similarly, require from relations implied by TRC-expressions, to be compatible with some relation scheme of the corresponding ER-schema.

Definition 4.2 - Attribute Compatibility.

Let \((R, K, L)\) be the relational schema of ER-consistent database, and let \( \Psi \) be an TRC-expression over \( R \). \( \Psi \) is said to be compatible with \( R \) iff \( A_i \land A_i \).

Note that the compatibility condition is stronger than for traditional relational databases, by not allowing the multiple appearance of an attribute in the header of the TRC-expression. The applicability of TRC in ER-consistent databases is embodied by the compatibility condition alone; although no more than an attribute compatibility, the condition guarantees the ER-consistency of the expression. The only modification required in order to give TRC an ER-orientation is a notational one, presented below.

Definition 4.3 - Entity-Relationship Oriented TRC (TRC\(_{ER}\)).

Let \( (R, K, I) \) be an ER-consistent relational schema, corresponding to ERD \( G_{ER} \); TRC\(_{ER}\) over \( (R, K, I) \), is defined after TRC as follows (only the additions are mentioned):

- terms an indexed variable can be also of the form \( x[E_{K_i}] \), where \( x \) is a tuple variable with range \( R_i \), and \( E_{K_i} \subseteq E_{K_i} \).
Predicates indexed variable of the form \( x[EK_i] \) can appear only in binary-comparison predicates of the form \( x[EK_i] \Theta y[EK_j] \), where \( \Theta \) is either \( = \) or \( \neq \).

Expression there is some \( R_i \in R \) such that \( TRC_{ER} \)-expression is compatible with \( R_i \).

\( TRC_{ER} \) can be further modified and made to use directly the ER-structures, instead of its relational translates. The existence of a relation-scheme with which the \( TRC_{ER} \)-expression can be associated, will be later relaxed and replaced by the existence of a transformation resulting in the addition of such a relational-scheme to the respective schema.

An Entity-Relationship Calculus (ERC) has been proposed in [AC]. Although inspired by the relational calculus, it is not clear how ERC relates to it, that is, what is the power of ERC. Moreover, while explicitly discarding the comparison of unrelated entity/relationship-sets, [AC] is not concerned whether ERC-expressions imply well-defined ER-structures. In this sense proposals of ER algebras ([MR], [CCÉ], [PS]) do not much better.

V. INCREMENTAL STATE MODIFICATION IN ER-CONSISTENT DATABASES.

We investigate in this section the behavior of ER-consistent databases under schema-invariant database changes. Our main interest is in state modifications that are incremental and that keep invariant the set of all the entities in the database.

**Definition 5.1 - State Modification.**

Let \( r \) be a database state associated with some relational schema, \( (R', K, I) \). A schema-invariant mapping of \( r \) into \( r' \), is called state modification.

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. Let relation \( r_i \) be associated with relation scheme \( R_i \), the translate of an e-vertex/r-vertex \( X_i \). An elementary update in a relational database may be: (i) modify attribute value in a tuple; (ii) delete tuple from relation; and (iii) insert tuple into relation. Updates in an ER-consistent relational database refer to information, rather than data, structures. An elementary update refers to an entity/relationship, or an attribute of an entity/relationship, and embody the enforcement of the existence constraints specified by the ERD edges.
Definition 5.2 - Locally Performed Update.
Let \((R, K, I)\) be an ER-consistent relational schema, \(R_i\) a relational scheme of \((R, K, I)\), and \(r_i\) the relation associated with \(R_i\). The deletion/insertion of a tuple, \(t\), from/into \(r_i\), is locally performed as follows:

(insert) \(r_i := r_i \cup \{t\}\); and \(\forall R_j\), such that \(R_j \subseteq R_i \in I\) : \(r_j := r_j \setminus \{t\} \cup \{t' \in r_j \) and \(t'[\{EK_j\}] = t[\{EK_j\}]\).

(delete) \(r_i := r_i \setminus \{t\}\); and \(\forall R_j\), such that \(R_j \subseteq R_i \in I\) : \(r_j := r_j \setminus \{t\} \cup \{t' \in r_j \) and \(t'[\{EK_j\}] = t[\{EK_j\}]\), where * specifies the concatenation of \(t[\{EK_j\}]\) with all the missing values corresponding to the attributes of \(A_j \in \{EK_j\}\).

Proposition 5.1
Let \((R, K, I)\) be an ER-consistent relational schema, \(R_i\) a relational scheme of \((R, K, I)\), and \(r_i\) the relation associated with \(R_i\), and \(J_i \subseteq I\), the subset of inclusion dependencies involving \(R_i\). The locally performed deletion/insertion of a tuple, \(t\), from/into \(r_i\), is locally ER-consistency preserving, that is, all the inclusion dependencies of \(J_i\) are preserved.

Proof: straightforward.

Updates for which global ER-consistency preservation is equivalent to the local ER-consistency preservation, are called incremental.

Definition 5.3 - Incremental Tuple Update.
Let \((R, X, I)\) be an ER-consistent relational schema, \(R_i\) a relational scheme of \((R, X, I)\), and \(r_i\) the relation associated with \(R_i\). The insertion/deletion of a tuple, \(t\), into \(r_i\), is incremental iff

(insert) \(\forall R_j, R_i \subseteq R_j \in I : \exists t' \in r_j, t'[\{EK_j\}] = t[\{EK_j\}]\).

(delete) \(\forall R_j, R_i \subseteq R_j \in I : \exists t' \in r_j, t'[\{EK_j\}] = t[\{EK_j\}]\).

For non-incremental updates, the procedure performing the deletion/insertion of a single tuple from/into some relation \(r_i\), is more complex: let \(r_i\) be associated with the relational translate of an entity-set/relationship-set \(X_i\) of a reduced ERD \(G'_{ER}\), and let \(G_{ER}(X_i^{\text{update}})\) be one of the following acyclic subgraphs of \(G'_{ER}\) (figure 3):

\[
G_{ER}(X_i^{\text{delete}}) = (V_i, H_i) : V_i = X_i \cup \{X_j | X_j \rightarrow X_i \in G'_{ER}\}, \quad H_i = \{X_k \rightarrow X_j | X_k, X_j \in V_i, X_k \rightarrow X_j \in G'_{ER}\};
\]

\[
G_{ER}(X_i^{\text{insert}}) = (V_i, H_i) : V_i = X_i \cup \{X_j | X_j \rightarrow X_i \in G'_{ER}\}, \quad H_i = \{X_k \rightarrow X_j | X_k, X_j \in V_i, X_k \rightarrow X_j \in G'_{ER}\}.
\]

It is easy to see that the procedure performing the deletion/insertion of a single tuple from/into \(r_i\), consists of the BFS-oriented (cf. [Even]) propagation of the update from \(r_i\) to some, or all, of the relations corresponding to the vertices of \(G_{ER}(X_i^{\text{update}})\).

We are concerned in the sequel of this paper only with state modifications which are incremental, that is...
consisting of propagation-free updates, and that keep invariant the set of entities in the database.

Definition 5.4 - Incremental State Modification, Conservative State Modification.

A state modification mapping \( r \rightarrow r' \), is called

(i) **incremental** iff \( r \) and \( r' \) differ in at most one relation;

(ii) **(entity) conservative** iff it does not involve insertions of tuples representing new entities.

New entities are introduced via entity-sets that are not specializations (subsets) of other entity-sets. Consequently, conservative state modifications are related only to relations corresponding either to relationship-sets or entity-subsets, characterized below.

**Proposition 5.2**

Let \( (R', K, J) \) be relational schema translate of an ERD, \( G_{ER} \); \( X_i \in G_{ER} \) an e-vertex/r-vertex, and \( R_i \) the relation scheme translate of \( X_i \). \( R_j \) is the translate either of an e-vertex representing a specialization entity-set, or an r-vertex iff \( \{ R_j \mid R_j \subseteq R_i \} \neq \emptyset \), and \( EK_i = \bigcup_{R_j \subseteq R_i} EK_j \).

Proof: following the ERD to relational schema mapping of section 3.

Expectingly, the ranges of the tuples that can be incrementally inserted/deleted from/into some relation depend on the inclusion dependencies involving the relation.
Definition 5.4 - Update Ranges.

Let \((R, K, I)\) be an ER-consistent relational schema, \(R_i\) a relational scheme of \((R, K, I)\), such that \(\{R_j \mid R_i \subseteq R_j \in I\} \neq \emptyset\), and \(r_i\) the relation associated with \(R_i\). The delete \((R_i^{\text{delete}})\) and insert \((R_i^{\text{insert}})\) ranges associated with \(R_i\) are following relations: 

\[
\begin{align*}
A_i^{\text{insert}} &= \bigcup_{R_j \subseteq R_i \in I} r_j[EK_j] \\
\hat{A}_i^{\text{delete}} &= \bigcap_{R_j \subseteq R_i \in I} r_j[EK_j]
\end{align*}
\]

Note: \(\hat{\wedge}\) denotes the algebraic natural join: 

\[
\begin{align*}
\wedge_{r_j [EK_i]} &= \{ t \mid \forall j : t[EK_i] \in r_j[EK_j] \}
\end{align*}
\]

Proposition 5.3

Let \((R, K, I)\) be an ER-consistent relational schema, \(R_i\) a relational scheme of \((R, K, I)\), and \(r_i\) the relation associated with \(R_i\). (i) The insertion/deletion of a tuple, \(t\), into/from \(r_i\) is incremental if the tuple \(t[EK_i] \in r_i^{\text{insert}} / t[EK_i] \in r_i^{\text{delete}}\). (ii) The composition of two incremental insertions/deletions into/from some relation, is also incremental.

Proof: straightforward.

VI. INCREMENTAL STATE MODIFICATION OPERATIONS.

We propose in this section two operations performing incremental and conservative state modifications. Without loss of generality, we shall assume that the target of such operations, \(R_i\), has \(A_i = EK_i\). For the sake of simplicity, we make the additional assumption that \(K_i = EK_i\) (meaning that there are no unary association cardinalities); the latter assumption allows us not to deal momentarily with the preservation of key dependencies.

Definition 6.1 - Increment and Decrement Relation.

Let \((R, K, I)\) be an ER-consistent relational schema, \(r\) the associated database state, \(R_i\) a relational scheme of \((R, K, I)\), and \(r_i\) the relation associated with \(R_i\). Let \(\Psi\) be an \(\text{TRC}_{ER}\) expression, compatible with \(R_i\), and evaluating to \(\psi\) over \(r\), such that \(\psi - r_i^{\text{insert}} / \psi \cap r_i \subseteq r_i^{\text{delete}}\).

The operations of increment/decrement with operand \(R_i\), are defined as follows:

- **prerequisite** (increment) \{ \(R_j \mid R_i \subseteq R_j \in I\) \neq \emptyset and \(EK_i = \bigcup_{R_j \subseteq R_i \in I} EK_j\) \};

- **syntax** \(\text{Increment}_{\Psi}(R_i)\), \(\text{Decrement}_{\Psi}(R_i)\);

- **semantics** \(r_i\) is mapped into \(r_i^{\prime}\): \(r_i^{\prime} := r_i \cup \psi\), and \(r_i^{\prime} := r_i - \psi\), respectively.

When \(\Psi\) is absent, increment and decrement are called maximization and minimization, respectively, and have the following syntax and semantics:
syntax \( \text{Max}(R_i), \text{Min}(R_i); \)

semantics \( r_i \) is mapped into \( r'_i : r'_i = r_i \cup r^\text{insert}_i \) and \( r'_i = r_i - r^\text{delete}_i \), respectively.

We shall now exemplify the use of increment and decrement operations in ER-consistent databases. Let \((R, K, I)\) be an ER-consistent schema translate of ERD, \(G_{ER}^r\), and \(X_i \in G_{ER}^r\), the e-vertex/r-vertex corresponding to the increment and decrement operand. The header of \( \Psi \) will be omitted whenever it is implied by the context.

(i) The union of a set, \( S \), of ER-compatible entity/relationship-sets is accomplished by minimization, provided \( \forall X_j \in S : X_j \rightarrow \bar{X}_j \in G_{ER}^r \). For instance, in figure 4, \( \text{Min}(\text{COURSE}) \) makes \( \text{COURSE} \) to be the union of \( \text{CS\_COURSE} \) and \( \text{EE\_COURSE} \), and \( \text{Min}(\text{T&E}) \) makes \( \text{T&E} \) to be the union of \( \text{TEACH} \) and \( \text{ENROLL} \).

(ii) The intersection of a set, \( S \), of ER-compatible entity-sets/relationship-sets is accomplished by maximization, provided \( \forall X_j \in S : \bar{X}_j \rightarrow X_i \in G_{ER}^r \). For instance, in figure 4, \( \text{Max}(\text{T&E}) \) makes \( \text{T&E} \) to be the intersection of \( \text{TEACH} \) and \( \text{ENROLL} \).

(iii) The selection is embedded in the increment and decrement operations. For instance, in figure 4, assume that the relation associated with \( \text{CS\_COURSE} \) is empty following, for instance,

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{example_diagram.png}
\caption{Entity-Relationship Diagram Example.}
\end{figure}
then the selection of 'COURSE entities with SUBJECT = 'CS' is accomplished by
\( \text{Increment}_\Psi(\text{CS\_COURSE}) \), where \( \Psi = \text{COURSE}[x] \land [\text{SUBJECT}] = 'CS' \).

(iv) The difference of two ER-compatible entity/relationship-sets can be accomplished through both increment and decrement operations. For instance, in figure 4, assume that the relation associated with EE\_COURSE has been maximized by \( \text{Max}(\text{EE\_COURSE}) \); then the difference between the set of all courses, presently in EE\_COURSE, and the set of courses with SUBJECT = 'CS', is accomplished by \( \text{Decrement}_\Psi(\text{EE\_COURSE}) \), where 'ψ' is as in (iii).

(v) The natural join like composition of entity/relationship-sets can be accomplished by maximization, provided the required structure pattern is satisfied. For instance, E&T in figure 4 represents the natural join of ENROLL and TAKE if-accompanied by \( \text{Max} \) (E&T).

The power of relational data manipulation languages is characterized by its completeness [CH]. The lower bound is the Codd-completeness, which means the language is expressive precisely as the first-order relational calculus. Similarly, we shall use \( \text{TRC}_{ER} \) as the completeness measure for conservative state modifications operations within ER-consistent databases.

Definition 6.2 - \( \text{TRC}_{ER} \)-completeness.
A set of operations, \( \Omega \), performing conservative state modifications, is said to be \( \text{TRC}_{ER} \)-complete iff, given an ER-consistent database, \( r \), associated with schema \((R, K, I)\), and any \( R_i \in R \), for any \( \text{TRC}_{ER} \)-expression, \( \Psi \), compatible with \( R_i \), there is a sequence of \( \Omega \) operations, that map \( r \) into a consistent database state \( r' \), such that \( r' = r \cup \Psi / r_i - \Psi \).

Proposition 6.1
The Increment and Decrement operations form an \( \text{TRC}_{ER} \)-complete set of incremental and conservative state modification operations.

Sketch of the proof: Let \( r \) be an ER-consistent database associated with schema \((R, K, I)\), and \( \Psi \) an \( \text{TRC}_{ER} \)-expression, compatible with \( R_i \in R \), and evaluating to \( \Psi \) over \( r \). \( \text{G}_{ER} \) is the ERD corresponding to \((R, K, I)\).

(i) Increment/Decrement operations are \( \text{TRC}_{ER} \)-bounded, by definition;

(ii) Increment/Decrement operations are \( \text{TRC}_{ER} \)-expressive: for Increment, the case to be proved is for \( \delta = (\Psi - r_i) - r_i \cup \Psi \), since the insertion of \( \delta \) in \( r_i \) will propagate to the relations corresponding to the vertices of \( \text{G}_{ER}(X_i^{\text{inns}}) \) (see section 5); the propagation, over the acyclic
ERD subgraph, defines a certain order, $O_i$, for the vertices of $G_{ER}(X_i^{\text{inew}})$; for every vertex $X_j \in G_{ER}(X_i^{\text{inew}})$ the propagation consists of the insertion of $\psi_j = \delta[EK_j]_j$ into $r_j$ (padded with null values for $A_j = \bar{EK}_j$); then the sequence consists of $\text{Increment}_{\psi_j}(R_j)$ for all vertices $X_j \in G_{ER}(X_i^{\text{inew}})$, ordered according to the inverse of $O_i$, and ending with $\text{Increment}_{\psi}(R_i)$; for $\text{Decrement}$ the proof is similar, but using the propagation over $G_{ER}(X_i^{\text{inew}})$.

VII. INCREMENTAL SCHEMA RESTRUCTURING.

Schema restructuring is part of both database design and database reorganization. For relational schemas of the form $(R, K, I)$, the basic schema restructuring manipulations are the addition and removal of relation schemes, together with the adjustment of key and inclusion dependencies. However, adding and removing relations are just expressions of information structure specification and evolution, and as such must have information structure (ERD) transformations counterparts. We are briefly reviewing incremental restructuring of relational ER-consistent schemas, following [MM].

Definition 7.1 - ER-Vertex Transformation.

An ERD transformation consisting of a vertex connection/disconnection, which maps a well-formed ERD into another well-formed ERD is called an ER-vertex transformation.

In [MM] we have proposed a set of ER-vertex transformations. Examples of such transformations, without syntactic details, are given in figure 5. Given an ER digraph its relational interpretation is given by $T_r$, while the mapping of the ER-vertex transformations to relational restructuring manipulations is denoted $T_{\text{min}}$:

$$
\begin{align*}
G_{ER} \xrightarrow{T_r} T_r & \xrightarrow{\text{ER-Vertex Transformation} (T)} \xrightarrow{T_{\text{min}}} G'_{ER} \\
(R, K, I) \xrightarrow{T_r} & \xrightarrow{T_{\text{min}}} (R', K', I')
\end{align*}
$$

The connection/disconnection of an e-vertex/r-vertex is mapped to the addition/removal of the corresponding relation scheme translate. Informally, such an addition/removal implies the addition/removal of the associated key, and the inclusion dependencies involving the relation scheme. The relation scheme addition includes also the removal of additional inclusion dependencies in order to preserve the ER-consistency of the schema, while the relation scheme removal includes the addition of the inclusion dependencies whose
Fig. 5. (1) Connect EMPLOYEE; Connect CITY; Connect WORK
(2) Disconnect WORK; Disconnect CITY; Disconnect EMPLOYEE

Implication have depended on the removed relation. A basic characteristic of schema restructuring is incrementality; schema restructuring 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.

Definition 7.2 - 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: \(R' := R \cup R_i\), \(K' := K \cup K_i\), and \((I' \cup K')^* = (I \cup K \cup I_i \cup K_i)^*\);

Removal: \(R' := R - R_i\), \(K' := K - K_i\), and \((I' \cup K')^* = ((I \cup K)^* - I_i - K_i)^*\).
We have shown in [MM] that for every ER-vertex transformation, \( \tau_i \), \( \sigma_i = T_{\text{sen}}(\tau_i) \), is incremental. While incrementality characterizes one-step schema modifications, reversibility assures that every such modification can be undone in one step: 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. In [MM] we have shown that the set of ER-vertex transformations is complete in the following sense.

**Definition 7.3 - 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, 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).

For non-empty states, the schema restructuring has the following effect on the database state.

**Definition 7.4 - Effect of Schema Restructuring on the Database State.**
Let \( R' \) be an ER-consistent database associated with schema \( (R, K', I') \), and \( R_i \) a relational scheme. The effect of the addition/removal, \( \sigma_i \), of \( R_i \) to/from \( (R, K', I') \), \( \sigma_i^{-1} \), is defined follows:

**removal** delete the relation associated with \( R_i \);

**addition** associate with \( R_i \) a relation with an empty delete range, \( r_i := \bigcup_{j \subseteq R_i} r_j[EK_j] \).

While the relational scheme removal evidently maps the database state to a consistent new state, the addition, for \( r_i \neq \emptyset \), could trigger an insert propagation from \( R_i \) to relations associated with the translates of the vertices of \( G_{ER}(X^{\text{new}}) \). The following proposition gives the condition that guarantees incrementality for state modifications induced by the addition of relational schemes.

**Proposition 7.1 [MM]:**
Let \( (R, K, I) \) be an ER-consistent relational schema, mapped, by the addition of \( R_i \), into \( (R', K', I') \). The addition of \( R_i \) maps the database state to a consistent state iff \( \forall R_m \in \{R_i \mid R_i \preceq R_j \in I'\}, \forall R_k \in \{R_k \mid R_k \preceq R_i \in I'\}, R_m[EK_m] \subseteq r_m[EK_m] \).

Proof: straightforward.
The evolution of an information system is reflected in data modeling by what is called **database reorganization** (TL). Database reorganization consists of schema restructuring coupled with entity-conservative state modification. Since algebraic operations consist of the embedding of schema restructuring and state modification, relational database reorganization has been mostly centered on relational algebra (e.g. [ST]). Similarly, for ER-consistent databases reorganization consists of schema restructuring coupled with compatible, entity-conservative, state modifications. We shall denote by \( \Sigma \) and \( \Omega \) any sets of restructuring manipulations and conservative state modification operations, respectively, and by \( \Sigma^{ER} \) and \( \Omega^{ER} \) the corresponding sets of incremental operations, proposed by us. First we shall define the compatibility of a reorganization manipulation and a state modification operation.

**Definition 8.1 - Restructuring Manipulation and State Modification Operation Compatibility.**

Let \( \Sigma \) and \( \Omega \) be sets of restructuring manipulations and state modification operations, respectively. A pair \( (\sigma, \omega) \), where \( \sigma \in \Sigma \) and \( \omega \in \Omega \), is said to be **compatible** over an ER-consistent database with schema \( (R, K, I) \), corresponding to ERD \( G_{ER} \), iff either (i) \( \omega \) is empty (null state transformation); (ii) \( \sigma \) is empty (null restructuring) and \( \omega \) has as argument some \( R_i \in R \); or (iii) \( \sigma \) results in the addition of relation-scheme \( R_i \) to \( R \), where \( R_i \) is the relational translate of a vertex representing an entity-subset or relationship-set, and \( \omega \) is insertion-oriented, and compatible with \( R_i \).

It is easy to check that any other combination of restructuring manipulation and state modification operation cannot be compatible. For instance a delete-oriented state modification cannot be coupled with the addition of a new relation-scheme, because the associated relation has an empty delete range.

**Definition 8.2 - Incremental Database Reorganization Operation.**

Let \( (R, K, I) \) be an ER-consistent relational schema, associated with database state \( r \). A **database reorganization operation** is a compatible pair \( (\sigma, \omega) \), of a restructuring manipulation, \( \sigma \), and an entity-conservative state modification operation, \( \omega \):

\[
\begin{align*}
\text{(R, K, I)} \quad \sigma \quad (R', K', I') \\
\downarrow \quad \sigma' \quad \downarrow \quad \omega \quad \downarrow \\
r \quad \sigma' \quad \omega \quad r'
\end{align*}
\]

A database reorganization operation, \( (\sigma, \omega) \), is said to be **incremental** iff (i) \( \sigma \) is an incremental restructuring manipulation; and (ii) \( \sigma' \) and \( \omega \) are incremental state modifications (recall that not all state
modifications induced by restructuring manipulations are incremental).

**Definition 8.3 - Database Reorganization Operation Completeness.**

Let $\Sigma$ be a set of restructuring manipulations, $\Omega$ a set of entity-conservative state modification operations, and $\Lambda$ a set of reorganization operations, \{ $(\sigma, \omega) \mid \sigma \in \Sigma, \omega \in \Omega, \sigma$ and $\omega$ compatible \}. $\Lambda$ is said to be complete iff given an ER-consistent database $r$ with schema $(R, K, I)$, and any reorganization, $\rho$, consisting of the compatible coupling of (i) ER-consistency preserving addition/deletion of $R_i$ to/from $(R, K, I)$, and (ii) the insertion/deletion into/from $R_i$ of $\psi$, where $\psi$ is the evaluation over $r$ of TRC$_{ER}$-expression $\Psi$, compatible with $R_i$; there exists a sequence of $\Lambda$ operations performing $\rho$.

**Theorem 6.2.**

Let $\Lambda^{ER}$ be the set \{ $(\sigma, \omega) \mid \sigma \in \Sigma^{ER}, \omega \in \Omega^{ER}, \sigma$ and $\omega$ compatible \}. $\Lambda^{ER}$ is complete.

Sketch of the proof:

(i) for $\rho$ consisting of either restructuring, or state modification, alone, the respective completeness of $\Sigma^{ER}$ and $\Omega^{ER}$ are enough;

(ii) for $\rho$ consisting of the coupling of an incremental restructuring with a state modification, if the state modification induced by the restructuring is incremental, then, again, the completeness of $\Omega^{ER}$ is enough; otherwise the reorganization has to be preceded by a state modification assuring the state-incrementality condition of proposition 7.1;

(iii) the only meaningful non-incremental relation addition is that of the relational translate of a vertex representing an entity-set [MM], which is not compatible with an entity-conservative state modification.

**IX: CONCLUSION**

Database reorganization is an expression of the evolution of an information system. Since the capability of relational databases to model information oriented systems is expressed by ER-consistency, we have investigated the relational database reorganization in an ER-consistent environment. Database reorganization operations have been defined as consisting of compatible pairs of restructuring and entity-conservative state modification manipulations. Recently we have defined a set of incremental and reversible schema restructuring manipulations as the translates of a set of vertex-oriented ER-diagram transformations. In the present paper we have proposed two operations that perform incremental and entity-conservative state modifications, and have proved their relational-calculus based completeness. The proposed set of database
reorganization operations is shown to be complete.

REFERENCES.


