INCREMENTAL REORGANIZATION OF RELATIONAL DATABASES

Victor M. Markowitz and Johann A. Makowsky

Department of Computer Science, Technion
Israel Institute of Technology, Haifa 32000, Israel

Abstract The evolution of an information system is reflected in data modeling by *database reorganization. Entity-Relationship(ER) consistency* expresses the capability of relational databases to model information oriented systems. A relational schema consisting of relation schemes, together with key and inclusion dependencies, is said to be ER-consistent if it complies with an entity-relationship structure, meaning that it is representable by an ER-Diagram. For ER-consistent schemas the basic restructuring manipulations are the addition and removal of relation schemes, coupled with the modification of the key and inclusion 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. For non-empty database states the schema restructuring manipulations must be associated with state mappings, and this leads us to the definition of database reorganization operations; *database reorganization operations* consist of *compatible* pairs of incremental restructuring manipulations and *entity-bounded* state mappings. For the specification of ER-consistent database state mappings, we propose an *Entity-Relationship Calculus.*

1. 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 state mapping. Since algebraic operations consist of the *embedding* of schema restructuring and state mapping, 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 mappings are 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 [Mar] we have defined a set of *incremental* and *reversible* schema restructuring manipulations as the translates of a set of vertex-oriented ERD-transformations. While incrementality characterizes the locality of one-step restructuring manipulations, reversibility assures that every such manipulation can be undone also in one step.

For non-empty database states the schema restructuring manipulations must be associated with state mappings. This leads us to the definition of database reorganization operations; *database reorganization operations* consist of *compatible* pairs of incremental restructuring manipulations and *entity-bounded* state mappings. An important characteristic of database reorganization is its locality, captured by the concept of *reorganization incrementality* which combines the incrementality embodied by schema restructurings, with the incrementality of the associated state mappings.

We propose a calculus-oriented ER notation to express state mappings in ER-consistent databases. The *Entity-Relationship Calculus (ERC)* proposed by us is mainly an ER-oriented notational adaptation of the Tuple Relational Calculus, coupled with an ERC-expression/ERD-transformation *compatibility* condition. An *Entity-Relationship Calculus (ERC-AC)* has been proposed in [AC]. Although inspired by the relational calculus, it is not clear how ERC-AC relates to it, that is, what is the power of ERC-AC. However by explicitly discarding the comparison of unrelated entity/relationship-sets, ERC-AC is obviously less powerful than the relational calculus, and for a disputable reason for that matter. ERC-AC, as almost all the other ER-oriented languages, is query biased, so that no attention is paid to whether ERC-AC expressions imply well-defined ER-structures.

The paper is organized as follows. The next section introduces ER Diagrams. The concept of ER-consistency is reviewed in section 3. In section 4 we investigate state mappings in ER-consistent databases. Relational schema restructuring is briefly reviewed in section 5. In section 6, we define database reorganization operations. In section 7 we define the Entity-Relationship Calculus. In section 8 we discuss various ER-algebra proposals and show how our reorganization operations can be used to specify algebra-oriented operations.

2. Role Free Entity-Relationship Diagrams

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* groups atomic values of a certain type; value-sets are the direct correspondents of the relational domains with interpreted elements. A relationship represents the interaction of several entities, and relationships of the same type are grouped in a *relationship-set*. An attribute is associated with one or several value-sets. Attributes associated with the same collection of value-sets are said to have the same *type*. A subset of the attributes associated with an entity-set may be specified as the, not necessarily unique, *entity-identifier*. Entity-identifiers are used to distinguish among the occurrences of an entity-set. An entity-set in a relationship-set may have a *role*, expressing the function it plays in the relationship-set. *Association cardinality* constraints are restrictions on the 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 all the other entity-sets involved in the relationship-set.

ER-schemas are expressible in a diagrammatic form called *ER-diagram (ERD)* which we define as a directed graph (example in figure 1). Entity-sets, relationship-sets, and attributes of entity-sets or relationship-sets, are represented by entity, relationship and attribute vertices, respectively. Entity, relationship and attribute vertices, are denoted as a-vertices, r-vertices, and e-vertices, respectively, and represented graphically by circles, diamonds, and rectangles, respectively. ERD vertices are connected by directed edges represented graphically by arrows; edges connecting r-vertices are represented graphically by dashed arrows. Every vertex is labeled by the name of the associated entity-set, relationship-set, or attribute name; e-vertices and r-vertices are uniquely identified by their labels globally, while a-vertices are uniquely identified by their labels only locally, within the set of a-vertices connected to some e-vertex/r-vertex. The *reduced ERD* is an ERD with the a-vertices, and all their incident edges, removed.

We deal in our paper with ERDs without role and cardinality specifications, called *role-free ERD*s. A role free ERD does not allow, for instance, the association of entities from a same entity-set. A formal definition of role-freeness is given later (constraint ER5 of definition 2.2). Without any loss of generality, we also assume that relationship-sets have no attributes of their own.

Notations (1):

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

ERD edges specify existence constraints:

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

$(E_i \overset{ISA}{\rightarrow} E_j)$ the ISA relationship expresses a *subset* relationship between two entity-sets; E_i is said to be an *entity-subset (specialization)* of E_j, and E_j is said to be a *generic entity-set (generalization)* of E_i;

$(E_i \overset{ID}{\rightarrow} E_j)$ 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); E_i is said to be a *dependent* of E_j ;

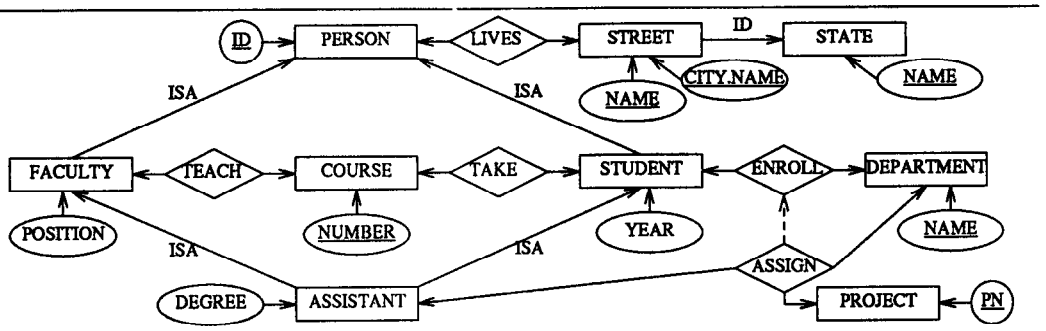
$(R_i \rightarrow E_j)$ relationship-set R_i involves entity-set E_j , therefore a relationship from R_i exists provided the related entity from E_j , also exists;

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

Notations (2):

- $Atr(E_i) \triangleq \{A_j \mid A_j \rightarrow E_i \in G_{ER}\}$, denotes the set of a-vertices connected to an e-vertex E_i ;
- $Id(E_i) \subseteq Atr(E_i)$, denotes the *entity-identifier* specified for e-vertex E_i;
- $GEN(E_i) \triangleq \{E_k \mid E_i \overset{ISA}{\rightarrow} E_k \in G_{ER}\}$, denotes the set of generalizations of entity-set E_i;
- $ENT(E_i) \triangleq \{E_k \mid E_i \overset{ID}{\rightarrow} E_k \in G_{ER}\}$, denotes the set of entity-sets on which entity-set E_i is ID-dependent;
- $ENT(R_i) \triangleq \{E_k \mid R_i \rightarrow E_k \in G_{ER}\}$, denotes the set of entity-sets associated by relationship-set R_i ;
- $ENT \rightarrow\!\!\rightarrow ENT'$ denotes the existence of an 1-1 correspondence, C , between the e-vertices of two sets of e-vertices, ENT and ENT' , belonging to an ERD, G_{ER} :

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



Note: ASSIGN → ENROLL means that an assistant is assigned to projects only in the departments he is enrolled in.

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

Definition 2.1 - Specialization Cluster.
Let G_{ER} be an ERD and $E_i \in G_{ER}$ an e-vertex; the *specialization cluster rooted* in E_i, $SPEC^*(E_i)$, is the set of all the e-vertices representing *specializations* of the entity-set represented by E_i :

$$SPEC^*(E_i) = E_i \cup \{E_j \mid E_j \xrightarrow[ISA]{} E_i \in G_{ER}\}.$$

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

Definition 2.2 - Role-Free Entity-Relationship Diagram.
A *Role-Free Entity-Relationship Diagram (ERD)* is a finite labeled digraph $G_{ER}=(V,H)$, where V is the disjoint union of three subsets of vertices: E (e-vertices), R (r-vertices), and A (a-vertices); H is the set of directed edges, where an edge can be of one of the following forms: $A_i \rightarrow E_j$, $E_i \rightarrow E_j$, $R_i \rightarrow E_j$, and $R_i \rightarrow R_j$. G_{ER}, obeys the following constraints:

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

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

(ER3) for any e-vertex/r-vertex X_i holds: $\forall (E_j, E_k) \in ENT^2(X_i)$: $\nexists E_m$ s.t. both $E_j \rightarrow\!\!\!\!\rightarrow E_m$ and $E_k \rightarrow\!\!\!\!\rightarrow E_m \in G_{ER}$;

(ER4) $\forall E_i \in G_{ER}$: if $GEN(E_i) \neq \emptyset$ then $Id(E_i)=\emptyset$; $ENT(E_i)=\emptyset$; and E_i belongs to a *unique maximal specialization cluster*; otherwise $Id(E_i) \neq \emptyset$;

(ER5) $\forall R_i \in G_{ER}$: $ENT(R_i) \geq 2$ and $\forall R_i \rightarrow R_j \in G_{ER}$: $\exists\, ENT \subseteq ENT(R_i)$ such that $ENT \rightarrow\!\!\!\!\rightarrow ENT(R_j)$.

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

Definition 2.3 - ER-Compatibility.
The entity-set and relationship-set compatibility have the following graph-oriented analogs: (i) two e-vertices, E_i and E_j, are said to be *ER-compatible* iff they belong to a same specialization cluster; and (ii) two r-vertices, R_i and R_j, are said to be *ER-compatible* iff there is a one-to-one correspondence, $Comp(R_i,R_j)$, of compatible e-vertices between $ENT(R_i)$ and $ENT(R_j)$: $Comp(R_i,R_j)=\{(E_k,E_m) \mid E_k \in ENT(R_i), E_m \in ENT(R_j), E_k$ and E_m are ER-compatible$\}$.

Note that role-freeness assures the *uniqueness* of this later correspondence, whenever it exists.

3. Entity-Relationship Consistent Relational Databases

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

Definition 3.1 - Functional Dependency, Key, Key Graph.

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

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

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

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

Definition 3.2 - Inclusion Dependency, Properties, Graph.

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

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

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

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

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

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

Relations are manipulated by *relational algebra (RA)* operators (cf. [KS]): *union, intersection, difference, projection, selection, (natural) join,* and *cartesian product* . We use in this paper the RA union and natural join: let $R_i(A_i)$ and $R_j(A_j)$ be two relation schemes, associated with relations r_i and r_j respectively; t denotes a tuple, and $t[W]$ denotes the sub-tuple of t corresponding to attribute set W ;

union: $R_i \cup R_j \triangleq \{t \mid t \in r_i$ or $t \in r_j\}$;

join: $R_i \bowtie R_j \triangleq \{t[A_i \cup A_j] \mid t[A_i] \in r_i$ and $t[A_j] \in r_j\}$.

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

Proposition 3.1 (Proposition 4.1 [Mar]).
Let (R, K, I) be an ER-consistent relational schema, the translate of the ERD G_{ER} , whose reduced ERD is G'_{ER} , and let G_I and G_K be the inclusion dependency and key graphs associated with (R, K, I), respectively. (i) G_I and G'_{ER} are isomorphic;
(ii) I is typed, key-based, and acyclic; and (iii) G_I is a subgraph of G_K .

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

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

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

(2) for every e-vertex/r-vertex X_i define recursively the following set of a-vertices: $Key(X_i) \triangleq Id(X_i) \cup_{X_i \to X_j \in G_{ER}} Key(X_j)$;

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

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

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

Fig.2 T_s : Mapping ER-Diagram Into Relational Schema.

Proposition 3.2 (Corollary 4.2 [Mar]).
Let (R,K,I) be an ER-consistent relational schema; an inclusion dependency $R_i[X] \subseteq R_j[Y]$ is implied by I iff either it is trivial, or $X=Y$ and there is a path from R_i to R_j in the associated IND graph.
Notation: typing and key-basing allow to denote an inclusion dependency of an ER-consistent database, $R_i[K_j] \subseteq R_j[K_j]$, as $R_i \subseteq R_j$.

Definition 3.3 - Existence Key.
Let (R,K,I) be an ER-consistent relational schema, and $R_i \in R$; the *existence key* of R_i , EK_i is defined as the union of all the keys associated with the relation-schemes to which R_i is related by an inclusion dependency: $EK_i \triangleq \bigcup_{R_i \subseteq R_j \in I} K_j$.

Proposition 3.3 (Corollary 4.4 [Mar]).
Let (R,K,I) be the relational schema translate of an ERD G_{ER};
(i) a relation scheme $R_i \in R$ is the translate of a vertex representing a relationship-set or an entity-subset iff $K_i = EK_i$; (ii) a relation scheme $R_i \in R$, is the translate of an e-vertex iff either $K_i \not\subset EK_i$; or $\forall R_j \in R$ such that $K_j \subseteq EK_i$: $K_j = K_i$ ($= EK_i$) .

4. Update Behavior of ER-Consistent Databases

Database schema-invariant state mappings are generally known as *updates*. In ER-consistent relational databases, every relation corresponds to an entity-set or relationship-set, and every tuple represents an entity or relationship respectively. An elementary update in a relational database consist of: (i) modifying an attribute value in a tuple; (ii) deleting a tuple from a relation; and (iii) inserting a tuple into a relation. Updates in ER-consistent relational databases refer to information, rather than data, structures; thus, an elementary update refers to an entity/relationship, or an attribute of an entity/relationship. Let r be the database state associated with an ER-consistent schema; ER-consistency for r means that r satisfies the associated key and inclusion dependencies.

Proposition 4.1
Let r be an ER-consistent relational database state associated with schema (R,K,I). r is *ER-consistent* iff
$\forall r_i \in r$: $\bigcup_{R_j \subseteq R_i \in I} r_j[K_i] \subseteq r_i[K_i]$ and $r_i[EK_i] \subseteq \bowtie_{R_i \subseteq R_j \in I} r_j[K_j]$.
Proof:
r is *ER-consistent* iff $\forall R_j$ $(R_j \subseteq R_i \in I)$: $r_j[K_i] \subseteq r_i[K_i]$ and $\forall R_j$ $(R_i \subseteq R_j \in I)$: $r_i[K_j] \subseteq r_j[K_j]$. The proposition follows directly from the definitions of relational union and natural join.

Definition 4.1 - Incremental Update.
Let r be an ER-consistent relational database state associated with schema (R,K,I), and r_i the relation associated with $R_i \in R$; the deletion/insertion of a tuple, t , from/into r_i , and mapping r_i and r into r'_i and r' respectively, is said to be *incremental* iff r' is ER-consistent.

In general updates are not incremental, therefore the state ER-consistency is preserved by performing additional updates, that is, non-incremental updates *propagate* in the database.

Definition 4.2 - Local Update Propagation.
Let (R,K,I) be an ER-consistent relational schema, and r_i the relation associated with $R_i \in R$. The non incremental update of a tuple t over r_i *locally propagates* as follows:

delete $\forall R_j$ $(R_j \subseteq R_i \in I)$: delete from r_j the set of tuples $Del_j^i = \{t' \mid t' \in r_j$ and $t'[K_i] = t[K_i]\}$;

insert $\forall R_j$ $(R_i \subseteq R_j \in I)$: insert into r_j the tuple $Ins_j^i = \{ (t_i[K_j])^* \}$ where * specifies the concatenation of $t_i[K_j]$ with all the missing values corresponding to the attributes of $A_j - K_j$.

Proposition 4.2
Let r be the database state associated with the ER-consistent relational schema (R,K,I), r_i the relation associated with $R_i \in R$, and $I_i \subseteq I$, the subset of inclusion dependencies involving R_i. The *local propagation* of a non-incremental deletion/insertion of a tuple, t , from/into r_i , maps r into a state that satisfies all the inclusion dependencies of I_i , and is *minimal*, that is, no proper subset of updates has this property.
Proof: straightforward.

The overall update propagation, which maps the database state into an ER-consistent state, consists of recursive local propagations. Let r be the database state associated with the ER-consistent relational schema (R,K,I), the translate of ERD G_{ER}; let $update(t,r_i)$ be a non incremental insertion/deletion of tuple t into/from relation r_i associated with $R_i \in R$, where R_i is the translate of e-vertex/r-vertex $X_i \in G_{ER}$. It is easy to see that the propagation of $update(t,r_i)$ consists of the spanning of *at most* all the relations associated with the relation schemes corresponding to the vertices of $G_{ER}(X_i^{update})$, defined below (example in figure 3).

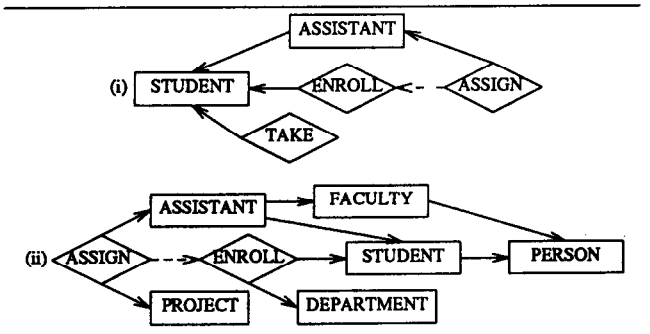


Fig.3 (i) $G_{ER}(\text{STUDENT}^{delete})$; (ii) $G_{ER}(\text{ASSIGN}^{insert})$.

Definition 4.3 - Update Propagation Subgraphs.
Let G_{ER} be an ERD, and X_i an e-vertex/r-vertex of G_{ER}; the *update propagation subgraph* induced by X_i, $G_{ER}(X_i^{update}) = (V_i, H_i)$, is one of the following acyclic subgraphs of the corresponding reduced ERD, G'_{ER} :

$$G_{ER}(X_i^{delete}): V_i = X_i \cup \{X_j \mid X_j \rightarrow\rightarrow X_i \in G'_{ER}\},$$
$$H_i = \{X_k \rightarrow X_j \mid X_k, X_j \in V_i,\ X_k \rightarrow X_j \in G'_{ER}\};$$
$$G_{ER}(X_i^{insert}): V_i = X_i \cup \{X_j \mid X_i \rightarrow\rightarrow X_j \in G'_{ER}\},$$
$$H_i = \{X_k \rightarrow X_j \mid X_k, X_j \in V_i,\ X_k \rightarrow X_j \in G'_{ER}\}.$$

Proposition 4.3
Let r be the database state associated with the ER-consistent relational schema (R, K, I), r_i the relation associated with $R_i \in R$. Any non-incremental deletion/insertion of a tuple, t, from/into r_i, can be accomplished by a sequence of *incremental* deletions/insertions.
Proof:
A non-incremental update over r_i propagates to the relations corresponding to the vertices of $G_{ER}(X_i^{update})$; the propagation, over the *acyclic* ERD subgraph, defines an order $<_i$ for the vertices of $G_{ER}(X_i^{update})$; for every vertex $X_j \in G_{ER}(X_i^{update})$ the propagation consists of deleting/inserting from/into r_j the set of tuples Del_j^i / Ins_j^i (definition 4.3); then the sequence consists of deleting/inserting from/into r_j, corresponding to every vertex $X_j \in G_{ER}(X_i^{update})$, in the order specified by the inverse of $<_i$, and ending with deletion/insertion of tuple t from/into r_i.

5. Incremental Schema Restructuring

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. Accordingly, ER-consistent relational schemas are suited for defining schema restructuring manipulations. We assume in this section that the database state is *empty*. The effect of schema restructuring manipulations on non empty database states will be investigated in the next sections. We briefly review incremental restructuring of relational ER-consistent schemas following [Mar].

Smooth schema restructuring, without major disruptions, is characterized by incrementality; informally, incrementality requires from a single manipulation to affect only *locally* the schema by keeping invariant the schema segment which is not in the immediate neighborhood of the manipulation. Accordingly, the effects of every single manipulation are easy to comprehend and manage. While incrementality characterizes one-step schema modifications, reversibility assures that every such modification can be undone in one step.

Definition 5.1 - Incremental and Reversible Schema Restructuring.
Let (R, K, I) be a relational schema mapped to (R', K', I') by an addition/removal restructuring manipulation, and let I_i be the subset of inclusion dependencies involving relation scheme R_i. A restructuring manipulation σ_i is said to be
(i) *incremental* iff [add (R_i)]: $R' = R \cup R_i$, $K' = K \cup K_i$, and $(I' \cup K')^+ = (I \cup K \cup I_i \cup K_i)^+$; [remove (R_i)]: $R' = R - R_i$, $K' = K - K_i$, and $(I' \cup K')^+ = ((I \cup K)^+ - I_i - K_i)^+$; and
(ii) *reversible* iff there is another restructuring manipulation, σ_j, such that the sequence of σ_i and σ_j applied on (R, K, I), returns the same schema, up to a renaming of attributes.

The major question with the restructuring of ER-consistent relational schemas is the preservation of ER-consistency and the specification of the ERD-transformation correspondent of every restructuring manipulation. We have proposed in [Mar] a set of *ERD-transformations*, Δ, consisting of connections/disconnections of vertices, and have specified the mapping of ERD-transformations into incremental and reversible restructuring manipulations. We have partitioned the set Δ of ERD transformations into three classes: (Δ_1) connection/disconnection of vertices representing entity-subsets and relationship-sets; (Δ_2) connection/disconnection of vertices representing entity-sets without dependent entity-sets, or representing generalizations of other entity-sets; (Δ_3) connection/disconnection of vertices representing conversions, of attributes into weak entity-sets, and weak entity-sets into independent entity-sets, together with their reverse conversions. For instance, the ERD of figure 4 is the result of transforming the ERD of figure 1, by connecting the following vertices: CITY, CS_DEPART, XX_DEPART, T/T, T&T, RX, T_COURSE, T&C, and TxT. Conversely, the ERD of figure 1 is the result of transforming the ERD of figure 4, by disconnecting the above vertices. Apart from the connection of CITY, which expresses

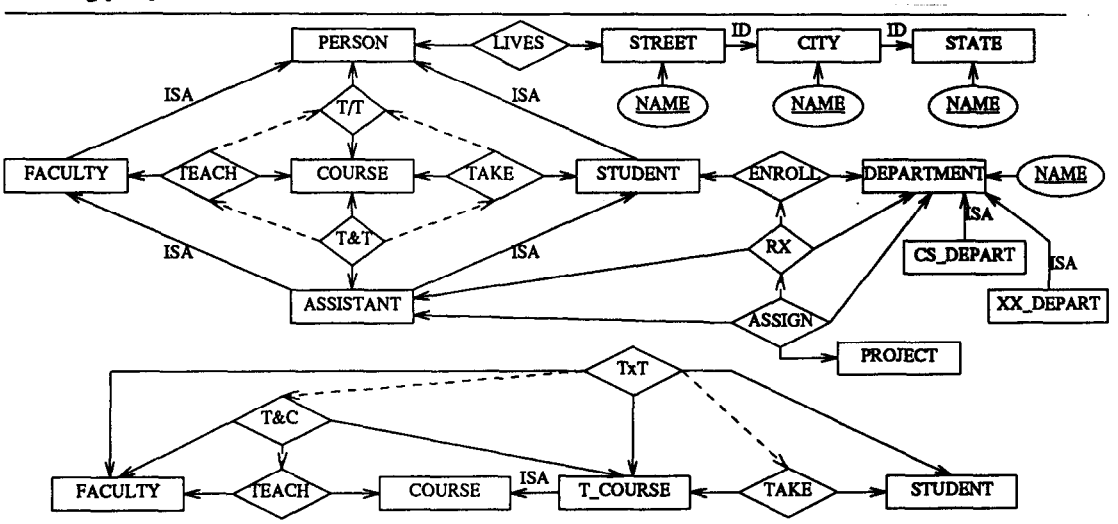
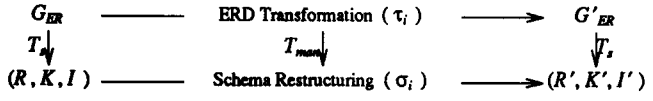


Fig.4 ERD-Transformation of the Entity-Relationship Diagram of Fig.1.

the conversion of an identifier attribute into a weak entity-set, transformation, all the other connections of this example are Δ_1 transformations.

Let (R, K, I) be the relational translate, by mapping T_s, of an ERD, G_{ER}; the mapping of the ERD-transformations into incremental and reversible restructuring manipulations, is defined by mapping T_{man}, as shown in the following diagram:



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 implication have depended on the removed relation. We have shown in [Mar] that for every ERD-transformation, τ_i, $\sigma_i = T_{man}(\tau_i)$ is incremental and reversible. In [Mar] we have shown that the set of ERD-transformations is *complete* in the following sense.

Definition 5.2 - ERD-Transformation Vertex-Completeness.
A set of ER-vertex transformations is said to be *vertex-complete* iff every 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).

6. Database Reorganization

The schema restructuring manipulations of the former section were under the assumption of empty database states. For non-empty database states the schema restructuring manipulations must be associated with state mappings; the association of incremental schema restructuring manipulations with state mappings is the basis of defining *database reorganization operations*. An important characteristic of database reorganization is its locality, captured by the concept of reorganization incrementality which combines the incrementality embodied by schema restructurings, with the incrementality of the associated state mappings. Reorganization state mappings must keep invariant the *identity* of the entities, and may introduce *new* entities into the database only by converting attribute values into entities, which is an Δ_3 ERD-transformation. This restriction is captured by the concept of *entity–boundness* defined below.

Definition 6.1 - Entity-Bounded State Mapping.
Let r be an ER-consistent database associated with schema (R, K, I) and which are mapped into (R', K', I') and r' respectively. The mapping of r into r' is said to be *entity-bounded* iff for every $R_i \in R'$, R_i the translate of an entity-set (not subset), either
(i) $R_i \in R$ and $r'_i \subseteq r_i$, or
(ii) $R_i \notin R$ and $r'_i[K_i] \subseteq \bigcup_{R_j \subseteq R_i \in I'} r_j[K_i]$.

Thus entity-bounded state mappings do not introduce new entities into existing entity-sets (i); any new *weak* entity-set resulting from the conversion of an attribute, consists initially of at most all the values of the converted attribute (ii); and any new *independent* entity-set is initially empty (ii).

Definition 6.2 - Database Reorganization.
Let r be an ER-consistent database associated with schema (R, K, I). σ and $\hat{\sigma}$ are a restructuring manipulation and a state mapping, defined over (R, K, I) and r respectively.

(i) σ and $\hat{\sigma}$ are said to be *compatible* iff either (a) σ is the null restructuring and $\hat{\sigma}$ is the replacement of r_i with r'_i; or (b) σ is the addition of relation-scheme R'_i to (R, K, I), and $\hat{\sigma}$ is the addition of r'_i to r; or (c) σ is the removal of relation-scheme R_i from (R, K, I), and $\hat{\sigma}$ is the removal of r_i from r.

(ii) An Ω *(database reorganization) operation* is a compatible pair $(\sigma_i, \hat{\sigma}_i)$ of an *incremental* restructuring manipulation σ_i, and an *entity-bounded* state mapping $\hat{\sigma}_i$, which maps (R, K, I) and r into (R', K', I') and r' respectively, such that r' is an ER-consistent database associated with (R', K', I').

(iii) An Ω–operation $(\sigma_i, \hat{\sigma}_i)$ is said to be *incremental* iff $\hat{\sigma}_i$ is an incremental state mapping.

(iv) A set of reorganization operations is said to be Ω*–complete* iff given an ER-consistent database r with schema (R, K, I), and any Ω–operation $(\sigma_i, \hat{\sigma}_i)$, there exists a sequence of reorganization operations performing $(\sigma_i, \hat{\sigma}_i)$.

Proposition 6.1
Let r be an ER-consistent database associated with schema (R, K, I), and let $(\sigma_i, \hat{\sigma}_i)$ be an Ω–operation which maps (R, K, I) and r into (R', K', I') and r' respectively.
(i) $\hat{\sigma}_i(r) = r - r_i$ is *incremental*;
(ii) $\sigma_i(r) = r[-r_i] \cup r'_i$ is *incremental* iff
$\bigcup_{R_j \subseteq R_i \in I'} r_j[K_i] \subseteq r'_i[K_i]$ and $r'_i[EK_i] \subseteq \bowtie_{R_i \subseteq R_j \in I'} r_j[K_j]$.

Proof:
(i) Insured by the definition of the removal restructuring [Mar].
(ii) The condition is from proposition 4.1; remains to prove that $\bigcup_{R_j \subseteq R_i \in I'} r_j[K_i] \subseteq \bowtie_{R_i \subseteq R_j \in I'} r_j[K_j]$, and this is insured by the specification of the addition restructuring (see [Mar]).

Without loss of generality we shall assume that any relation, R_i, affected by a reorganization operation is all key, that is, $A_i = K_i$.

Proposition 6.2
Let r be an ER-consistent database associated with schema (R, K, I), and $(\sigma_i, \tilde{\sigma}_i)$ an Ω–operation which maps (R, K, I) and r into (R', K', I') and r' respectively, such that $(\sigma_i, \tilde{\sigma}_i)$ is of one of the following forms:

(a) σ_i consists of the removal of R_i from (R, K, I) and $\tilde{\sigma}_i$ is the removal of r_i from r;

(b) σ_i consists of the addition of R'_i to (R, K, I) and $\tilde{\sigma}_i$ is the addition of $r'_i = \bigcup_{R_j \subseteq R_i \in I'} r_j[K_i]$ to r;

(c) σ_i is null and $\tilde{\sigma}_i$ is the replacement of r_i by $r'_i \supseteq r_i$;

(d) σ_i is null and $\tilde{\sigma}_i$ is the replacement of r_i by $r'_i \subseteq r_i$.

(i) Let $\tilde{\Omega}$ be a set of reorganization operations of the above forms. Then $\tilde{\Omega}$ is Ω*–complete*. (ii) Let $\tilde{\Omega}$ be a set of *incremental* reorganization operations of the above form. Then $\tilde{\Omega}$ is Ω*–complete*.
Proof:
It is enough to prove the Ω–completeness of $\tilde{\Omega}$. Let r be an ER-consistent database associated with schema (R, K, I), and $(\sigma_i, \hat{\sigma}_i)$ any Ω–operation which maps (R, K, I) and r into (R', K', I') and r' respectively.
For $\sigma_i =$ removal $\hat{\sigma}_i$ is incremental and of the form (a) above;
For $\sigma_i =$ addition, let G_{ER} be the ERD corresponding to (R, K, I), X_i the vertex corresponding to R'_i, and

$r_i^a = \bigcup_{R_j \subseteq R_i \in I'} r_j[K_i]$, $r_i^{in} = r'_i - r_i^a$, $r_i^{del} = r_i^a - r'_i$. The association of r'_i with R'_i can be done as follows: (1) associate r_i^a with R'_i; (2) insert r_i^{in} into r_i^a; (3) delete r_i^{del} from r_i^a.
Stage (1) corresponds to a state mapping of the form (b) above, and is incremental(proposition 6.1); stage (2) propagates to at most all the relations corresponding to the vertices of $G_{ER}(X_i^{insert})$, and can be accomplished by a sequence of incremental state mappings (proposition 4.3), which are of the form (c) above; and stage (3) propagates to at most all the relations corresponding to the vertices of $G_{ER}(X_i^{delete})$, and can be accomplished by a sequence of incremental state mappings (proposition 4.3), which are of the form (d) above.
For σ_i = null, the proof is similar.

Note that for an $\hat{\Omega}$–operation $(\sigma_i, \bar{\sigma}_i)$, where σ_i is null and refers to an entity-set translate, $\bar{\sigma}_i$ must be the replacement of r_i by $r'_i \subseteq r_i$ (entity-boundness).

7. Entity-Relationship Calculus

For schema restructuring manipulations, we have defined a complete set of ERD-transformations, having incremental schema restructuring mappings. Similarly, we propose a calculus-oriented ER notation to express state mappings.

Definition 7.1 - Entity-Relationship Calculus (ERC).
The *syntax* of ERC is defined as follows:

terms — *constants*, (entity or relationship) *variables*, or indexed variables; an *indexed variable* is of the form either $x[A]$ or $x[Y]$, where x is a variable, A is an attribute, and Y is an entity/relationship-set;

predicates — unary *range* predicates, associated with entity/relationship-sets, and having as arguments variables; binary *comparison* predicates, whose arguments are constants and indexed variables, of the form $x[A]$, such that A is an attribute of the entity-set associated with the range of x, and the attributes referred in the comparison are ER-compatible; and binary *equality-comparison* predicates, whose arguments are variables and indexed variables of the form $x[Y]$, such that the entity/relationship-sets referred in the comparison are ER-compatible, and if X is associated with the range of x, then X and Y are adjacent vertices in the ERD such that $X \rightarrow Y \in G_{ER}$;

propositions — either predicates, or of the form $P_1 \wedge P_2$, $P_1 \vee P_2$, $\neg P_1$, $P_1 \rightarrow P_2$, where P_1 and P_2 are propositions;

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

formulas — propositions, or quantified formulas of the form $(\exists x \in X)\Phi(x)$, and $(\forall x \in X)\Phi(x)$, meaning $(\exists x)(X(x) \wedge \Phi(x))$ and $(\forall x)(X(x) \rightarrow \Phi(x))$, respectively, where X is a range predicate involving x, x is free in Φ, Φ does not contain range predicates for x, and involves free variables other than x;

expression — $\{y_1 \cdots y_n \mid X_1(x_1) \wedge \cdots \wedge X_n(x_k) \wedge \Phi(x_1 \cdots x_k)\}$, where y_i are either variables or indexed variables, all referring to k different variables, x_i, X_i are range predicates, Φ is either absent or it is a formula with range-coupled quantifiers, without range predicates for $x_1 \cdots x_k$, and with $x_1 \cdots x_k$ its only free variables.

The power of relational data manipulation languages is characterized by their *completeness* [CH]. The lower bound is the *TRC-completeness*, which means the language is expressive *precisely* as the the first-order *Tuple Relational Calculus (TRC)*. We have defined ERC by adapting notationally the TRC as defined in [Pir]. In TRC the *terms* are constants, tuple variables, or indexed tuples, where an *indexed tuple* is of the form $t[A]$, with t being a tuple variable, and A an attribute; and *predicates* are 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. The mapping of an ERC expression into an TRC expression, is presented in figure 5. The mapping is straightforward and its correctness is guaranteed by the constraints put on the terms of form $x[Y]$ and the comparison predicates involving them. ERC allows the direct reference of the ER structures, but prevents the direct reference, within a relation, of the attributes belonging to existence keys. Consequently, ERC is trivially *TRC–complete*.

Input: G_{ER} and (R, K, I), an ERD and its relational schema translate; Ψ_{ER} an ERC expression over G_{ER};

Output: Ψ_R a TRC expression over (R, K, I);

terms — every *variable* x_i is mapped to tuple variable t_i;
every *indexed variable* $x_i[A_k]$ is mapped to tuple variable $t_i[A_k]$, where t_i is the mapping of x_i;
every *indexed variable* $x_i[X_j]$ is mapped to a set of tuple variables $\{ t_i[A_k] \mid A_k \in K_j$, where K_j is the key of the relational translate of X_j and t_i is the mapping of $x_i \}$;

predicates — every *range predicate* associated with entity/relationship-set X_i is mapped to a range predicate associated with R_i, the relational translate of X_i;
a *comparison predicate* of the form $x_i[A_k]\,\theta\,x_j[A_h]$ is mapped to $t_i[A_k]\,\theta\,t_j[A_h]$, where t_i and t_j are the mappings of x_i and x_j, respectively;
a *comparison predicate* of the form either $x_i\,\theta\,x_j[X_k]$, where x_i ranges over X_k, or $x_i[X_k]\,\theta\,x_j[X_k]$, is mapped to $\bigwedge_{A_m \in K_k} t_i[A_m]\,\theta\,t_j[A_m]$, where t_i and t_j are the mappings of x_i and x_j, respectively, K_k and K_h are the keys of R_k and R_h, the relational translates of X_k and X_h, respectively.

Fig.5 T_c : Mapping ERC Expression Into TRC Expression.

In a relational database, an TRC expression evaluates to a relation associated with some relation-scheme. For ER-consistent databases, we must also insure either the incremental addition of the new relation-scheme to the ER-consistent schema, or the existence of a relation-scheme with which the new relation would be associated. This leads us to the definition of the compatibility of ERC-expressions with ERD-transformations. The Ω–completeness of $\hat{\Omega}$ allows us to restrict the discussion to $\hat{\Omega}$–operations. Let r be an ER-consistent database associated with schema (R, K, I), and $(\sigma_i, \bar{\sigma}_i)$ an $\hat{\Omega}$–operation which maps (R, K, I) and r into (R', K', I') and r' respectively; let (R, K, I), (R', K', I'), and R_i be the relational translates of G_{ER}, G'_{ER}, and X_i respectively. Following proposition 6.2 we denote by Ψ_i^{min} the ERC-expression evaluating to $\bigcup_{R_j \subseteq R_i \in I'} r_j[K_i]$, and need to refer only to ERC-expressions associated with null ERD-transformations, and which specify state mappings that consist of the replacement of some relation r_i, associated with R_i, by r'_i such that either $r'_i \supseteq r_i$ or $r'_i \subseteq r_i$. Consequently, an ERC-expression Ψ_{ER} must evaluate, via T_c, to a relation that is either added to, or deleted from, an existing relation, that is $r'_i - r_i$ or $r_i - r'_i$ respectively, denoted Ψ_{ER}^{add}

and Ψ_{ER}^{del} respectively. Recall also that because of the entity-boundness condition, Ψ_{ER}^{add} can be associated only with ERD-vertices that represent either entity-subsets or relationship-sets. Actually the above separation allows us to impose the entity-boundness condition.

Definition 7.2 - ERC-Expression Δ-Compatibility.
Let G_{ER} be an ERD, τ_i an Δ–transformation referring to e-vertex/r-vertex X_i, Ψ_{ER} an ERC expression over G_{ER}, and $ENT(\Psi_{ER}) \stackrel{\Delta}{=} \{E_j \mid x$ appears in the header of Ψ_{ER} and E_j is either the range of x, or $E_j \in ENT(R_k)$, R_k is the range of $x\}$.

(i) Ψ_{ER}^{min} is said to be compatible with τ_i, where τ_i is not null, iff τ_i consists of the connection of X_i to G_{ER};

(ii) Ψ_{ER}^{add} is said to be compatible with τ_i, where τ_i is null and refers to $X_i \in G_{ER}$, iff

- X_i represents either an entity-subset or a relationship-set;
- $\forall (x_i, x_j)$ in the header of Ψ_{ER}^{add}, with ranges X_k and X_h respectively: X_k and X_h obey constraint (ER3); and

$[E_i]$ $ENT(\Psi_{ER}^{add}) = \{E_j\}$, and E_i is ER-compatible with $E_j \neq E_i$;

$[R_i]$ there is a 1-1 correspondence of ER-compatible e-vertices between $ENT(\Psi_{ER}^{add})$ and $ENT(R_i)$.

(iii) Ψ_{ER}^{del} is said to be compatible with τ_i, where where τ_i is null and refers to $X_i \in G_{ER}$, iff there is a single variable in the header of Ψ_{ER}^{del}, and its range is X_i.

The compatibility condition above, has the following relational correspondent: let (R, K, I), R_i, and Ψ_R be the relational translates of ERD G_{ER}, vertex X_i, and ERC-expression Ψ_{ER} over G_{ER}, respectively, and let $A_\Psi = \{A_{i_j} \mid A_{i_j} \in A_i$, $t_k[A_{i_j}]$ appears in the header of Ψ_R and R_i is the range of $t_k\}$. The compatibility of Ψ_{ER} with an ERD-transformation referring to vertex X_i implies $K_i = A_\Psi$. This condition is consistent with the fact that ER-compatibility corresponds in ER-consistent databases to key-identity [Mar]. Note that the above condition implies that multiple appearances of attributes in TRC-expression headers, are not allowed, which is a reflection of the ERD *role–freeness*. Note also that not every Ψ_R obeying the above condition has a vertex-compatible Ψ_{ER} correspondent.

8. Calculus Vs Algebra Oriented Database Reorganization

Tuple Relational Calculus (TRC) and Relational Algebra (RA) have been based originally on a table view of relations. Domain Relational Calculus (DRC) emerged from the attempt to offer an ER-oriented view of relations; in this view database domains roughly correspond to entity-sets, relations correspond to relationship-sets, and attributes express the role played by entities in relationships. Actually, the traditional relational model has not been rich enough to support a real ER-oriented view. It is worth noting that the ERC, presented in the previous section, maps straightforwardly to TRC rather than DRC, contrary to the believe that DRC is better suited to express ER-oriented semantics [Pir].

Following the acknowledgement of the fundamental weakness of the ER model, namely its lack of a well defined set of basic manipulations, several attempts have been made to define an ER-Algebra (ERA), starting with [MR] and followed by [PS] and [CCE]. The various ERA proposals have sought, more or less, correspondents to RA operations. All these proposals proved to be either inappropriate by being too close to the RA ([MR], [CCE]), or counter-intuitive [PS]. The simplicity and straight intuition of the ER concepts have been put aside in the search of analogies with the RA operations, and even RA-completeness, as in [CCE]. Take, for instance, the definition of set union and intersection. The result of the union(intersection) of two ER-compatible entity/relationship-sets is evidently a new entity/relationship-set, which is the smallest superset(greatest subset) of the operands. How the new entity/relationship-set relates to the operand entity/relationship-sets can be expressed by subset constraints, which are a special kind of existence constraints. The new entity/relationship-set inherits, or not, the attributes of the operand sets, *implicitly* as established by the obvious attribute inheritance rules in a subset hierarchy. Note that the subset constraints alone are not enough to represent properly the result of set difference, which would require some representation for the disjointness of compatible subsets. None of the above mentioned proposals have subset constraint representations. In [CCE] there exist no compatible entity-sets, and only relationship-sets can be combined to produce new relationship-sets that inherit explicitly all the attributes of the operands, almost as in RA. In [MR] the only improvement over [CCE] is the lack of explicit attribute inheritance. In [PS] operations are defined only over entity-sets (relationships are embedded into entities prior to any operation) and the explicit inheritance includes, besides attributes, also relationship-set involvements. All these proposals are based on an attribute-compatibility of entity/relationship-sets that reflect the RA attribute-compatibility, rather than an ER-compatibility. In the context of such definitions the RA-completeness of [CCE] seems to be a technical result without apparent practical significance.

One could wonder whether the lack of proper representations for subset, possibly other, constraints, is the only problem of defining an ERA. We believe that the answer is no. Excepting the set operations, it is hard to define operations analog to such RA operations as the projection and join, such that they would have some information-oriented meaning. Another major obstacle to an algebraic-oriented approach is the *nesting* of operations; it is very difficult, if not hopeless, to reach the generality of the RA composition, where any algebraic expression can be used as operand in any other algebraic operation, to any level of nesting. Assuming that all these problems are overcome, we are still left with the procedurality of an algebraic-oriented notation, overwhelming, we think, for a a high-level interface such as the ERD. Consequently, we doubt that there is any need for an RA-shaped ER notation. We shall show in the sequel of this section how RA-oriented manipulations can be specified with the database reorganization operations proposed by us. All examples refer to figure 4.

Let (R, K, I) and (R', K', I') be the relational translates of G_{ER} and G'_{ER}, and R_i the relational translate of e-vertex/r-vertex X_i. Let r be an ER-consistent database associated with schema (R, K, I), and $(\sigma_i, \tilde{\sigma}_i)$ an $\hat{\Omega}$ operation which maps (R, K, I) and r into (R', K', I') and r' respectively, such that:

- σ_i is the translate of an Δ_1 vertex connection (X_i represents an entity-subset or a relationship-set);
- $\tilde{\sigma}_i$ is the addition of either $\bigcup_{R_j \subseteq R_i \in I'} r_j[K_i]$ or $\bowtie_{R_i \subseteq R_j \in I'} r_j[K_j]$;

the ERC-expression specifying $\tilde{\sigma}_i$, is denoted $\Psi_i^{\min}$ in the former case, and $\Psi_i^{\max}$ in the later case. The ERD-transformations specifying σ_i, will be given without syntactic details.

Let *SET* be a set of ER-compatible entity/relationship-sets. The ***union*** of the entity/relationship-sets of *SET* is specified by the association of $\Psi_i^{\min}$ with the connection of an entity-subset/relationship-set X_i such that $\forall X_j \in SET : X_j \rightarrow X_i \in G'_{ER}$. For instance, the union of TEACH and TAKE is specified by (*Connect* T/T ; $\Psi_{T/T}^{\min}$).

The ***intersection*** of the entity/relationship-sets of *SET* is specified by the association of $\Psi_i^{\max}$ with the connection of an entity-

subset/relationship-set X_i such that $\forall X_j \in SET : X_i \rightarrow X_j \in G'_{ER}$. Note that the intersection of relationship-sets might be non incremental. For instance, the intersection of TEACH and TAKE is specified by (*Connect* $T\&T$; $\Psi^{max}_{T\&T}$) and propagates to ASSISTANT.

The *join* of two relationship-sets, or a relationship-set and an entity-set, is a generalization of the intersection. For instance, (*Connect* $T\&C$; $\Psi^{max}_{T\&C}$), specifies the join of TEACH and T_COURSE, while the join of T&C and TAKE is specified by (*Connect* TxT ; Ψ^{max}_{TxT}).

The *association* of entity-sets and relationship-sets results in new relationship-sets consisting of the cartesian product of the associated entity-sets/relationship-sets. We shall refer only to the association of entity-sets; the other cases are similar, although more complex. Let SET be a set of entity-sets, such that $\forall (E_i, E_j) \in SET$: E_i and E_j obey constraint (ER3). The *association* of the entity-sets of SET is specified by the association of Ψ^{max}_i with the obvious connection of a relationship-set R_i. For instance, the association of PERSON and COURSE is specified by (*Connect* T/T ; $\Psi^{max}_{T/T}$).

The *projection* of an entity-set/relationship-set on an entity-set results in a new entity-set, while the projection of a relationship-set on several entity-sets results in a new relationship-set. Let E_k represent an entity-set involved in relationship-set R_j. The projection of R_j on E_k results in a subset of E_k, E_i, specified by the association of Ψ^{min}_i with the corresponding connection. For instance, the projection of TAKE on COURSE is specified by (*Connect* T_COURSE ; $\Psi^{min}_{T\_C}$). Similarly, the projection of ASSIGN on ASSISTANT and DEPARTMENT is specified by (*Connect* RX ; Ψ^{min}_{RX}).

The *selection* of an entity/relationship-set, X_i, results in a subset of X_i, consisting of all the entities/relationships of X_i satisfying a certain condition. The selection is embedded, actually, in any reorganization operation. For instance, the selection of DEPARTMENT entities with NAME 'CS' is specified by (*Connect* CS_DEPART; $\{ x \mid DEPARTMENT[x] \wedge x[NAME]='CS' \}$).

We do not have a representation for constraints specifying the *disjointness* of two ER-compatible entity/relationship-sets. Consequently, the *difference* of two ER-compatible entity/relationship-sets can be expressed in a way similar to selection, rather than union or intersection. For instance, the difference of DEPARTMENT and CS_DEPART is specified by (*Connect* XX_DEPART; $\{ x \mid DEPARTMENT[x] \wedge CS\_DEPART[y] \wedge x \neq y \}$).

9. Conclusion

Database reorganization expresses 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 database reorganization in an ER-consistent environment. A natural extension of our work would be to incorporate more semantic modeling capabilities into the high-level ERD interface. Some of the possible extensions are listed below; all these extensions seem straightforward, but tedious:

- *Association cardinalities* have already been dealt with in [MMR], where unitary association cardinalities are mapped to functional dependencies and influence the specification of keys associated with the relational translates of relationship-sets.
- *Roles* express the functions played by entity-sets in relationship-sets. Roles are essential to distinguish different involvements of an entity-set in a same relationship-set, and could relax constraint (ER3) of the ERD definition.
- *Multivalued attributes* can be supported by *one-level nested relations* [FG], that is, relations with nesting done only over single basic attributes. Tuples in such relations consist of either atomic values, or sets of atomic values. The unnesting of such relations is straightforward. Assuming that identifier attributes are not multivalued, the mappings between ERDs and relational schemas are unchanged, since key and inclusion dependencies involve only identifier attributes, and the sets of restructuring manipulations and reorganization operations have to undergo only minor changes.
- *Disjointness constraints* specify the disjointness of ER-compatible entity/relationship-sets. for instance, disjointness constraints can express the *partitioning* of a generic entity-set into disjoint specialization entity-subsets. Disjointness constraints are supported in the relational model by *exclusion dependencies (EXD)* [CV].

References

[AC] P. Atzeni and P.P. Chen, "Completeness of query languages for the entity-relationship model" , *Entity-Relationship Approach to Information Modeling and Analysis*, P.P. Chen (ed), ER-Institute, 1981, pp. 111-124.

[CCE] D.M. Campbell, B. Czejdo, and D.W. Embley, "A relationally complete query language for an entity-relationship model", *IEEE 4th International Conference on Entity-Relationship Approach* , 1985, pp. 90-97.

[CH] A.K. Chandra and D. Harel, "Computable queries for relational data bases", *Journal of Computer and System Sciences* 21, 1980, pp. 156-178.

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

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

[FV] P.C. Fisher and D. Van Gucht, "Determining when a structure is a nested relation", Proceedings of teh Eleventh *VLDB Conference* , 1985, pp. 171-180.

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

[Mar] V.M. Markowitz, "Entity-relationship consistency for the relational model of databases", Ph.D. Thesis, Computer Science Department, Technion, Israel Institute of Technology, Haifa, June 1987.

[MR] V.M. Markowitz and Y. Raz, "An entity-relationship algebra and its semantic description capabilities", *Journal of Systems and Software* 2/3 ,4 (July 1984), pp. 147-162.

[PS] C. Parent and S. Spaccapietra, "An algebra for a general entity-relationship model", *IEEE Trans. on Software Engineering* 11 ,7 (July 1985), pp. 634-643.

[Pir] A. Pirotte, "High level data base query languages" in *Logic and Databases* , H. Gallaire and J. Minker (eds), Plenum Press, 1978, pp. 409-436.

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

[ST] B. Schneiderman and G. Thomas, "An architecture for automatic relational database system conversion", *ACM Transactions on Database Systems* 7 ,2 (Sep 1982), pp. 235-257.

[TL] D.C. Tsichritzis and F.H. Lochovsky, *Data models* Prentice-Hall, 1982.