数理解析研究所講究録
593巻 1986年 145-167

CLASSIFICATION AND TRANSFORMATIONS OF BINARY RELATIONSHIP RELATION SCHEMATA

Isamu KOBAYASHI

小林功武

SANNO Institute of Business Administration
School of Management and Informatics
Kamikasuya 1573, Isehara, Kanagawa 259-11, JAPAN

August, 1985

[ABSTRACT] In contrast to Relational Model that deals only with flat relations, models like Entity-Relationship Model, Network Model and Hierarchical Model make distinctions between entity relations and relationship relations. Distinctions between these two types of relations are usually perceived intuitively. However, to establish a formal basis of data translations among different data models, it seems necessary to describe such distinctions in a precise manner. In this paper, relationship relations are defined as those on which relationship rules hold. Binary relationship relations are of special importance because these can be regarded as correspondences and also as directed graphs. Cardinality of correspondences, and acyclicness and connectedness of directed graphs are important properties of binary relationship relations. With these properties taken into consideration a classification of binary relationship relation schemata can be established. Finally, four lossless transformations between two groups of binary relationship relation schemata are described. These play important roles in data translations among different data models and also in devising physical representation of binary relationship relations.

[KEYWORDS AND PHRASES] Classification of relationship relations, data model, data translation, relationship relation, schema transformation.

1. INTRODUCTION

Relational Model [COD 1970] deals only with flat relations. It is not aware of any distinction between entity relations and relationship relations. In contrast, models like Entity-Relationship Model [CHE 1976], Network Model [BAC 1974] and Hierarchical Model [TSI 1976] make distinctions between these two types of relations. In practical database design, such distinctions can often be perceived very intuitively. Therefore, many database besigners prefer the latter type of data models to the Relational Model, in particular, as a tool to be used in an early stage of database design. However, to establish a formal basis of data translations among various data models and hence among various database management systems, it seems necessary to describe these distinctions more precisely. In this paper, a formal definition of relationship relations in terms of relationship rules is presented first.

Among relationship relations, binary relationship relations are of special importance. This is because a binary relationship relation can be regarded as a correspondence also as a directed graph [KOB 1975a]. Cardinality of the correspondence, and acyclicness and connectedness of the directed graph are important properties. A classification of binary relationship relations in terms of several rules is presented next.

Finally four lossless schema transformations regarding binary relationship relation schemata are described. In a previous paper [KOB 1986], the author presented four basic lossless schema transformations that frequently appear in various parts of database theory. In relation to binary relationship relation schemata, four additional lossless schema transformations become necessary. These play important roles in data translations among data models and hence among database management systems, and also in devising physical representation of binary relationship relations.

For the four basic schema transformations (selection/union transformation, projection/natural-join transformation, encode/decode transformation and unset/set transformation) and rules for these transformations being lossless, readers are requested to refer the previous paper [KOB 1986].

2. RELATIONSHIP RULES AND RELATIONSHIP RELATIONS

First let us consider what relationships are. Let $\underline{P}$ be a relation schema corresponding to a three-position predicate symbol P. A tuple (u,v,w) in a relation P in $\underline{P}$ represents an atom P(u,v,w). Assume that a rule

$\forall u \forall v \forall w(P(u,v,w) \supset \Phi(v))$

holds, where $\Phi(v)$ is interpreted as v standing for a member of a power set of a Cartesian product. If a functional dependency

$\forall u \forall v \forall v' \forall w((P(u,v,w) \wedge P(u,v',w)) \supset v=v')$

holds, then an unnest transformation (a special unset transformation) defined by

$P(u,v,w) \wedge B(v,x) \rightarrow Q(u,x,w)$

is applicable to P where $B(v,x)$ is to be interpreted as x being a component of v. On the obtained relation schema Q, a rule

$\forall u \forall x \forall w(Q(u,x,w) \supset \Phi'(x))$

holds, where $\Phi'(x)$ is interpreted as x standing for a member of a Cartesian product. If a functional dependency

$\forall u \forall v \forall v' \forall w \forall w'((P(u,v,w) \wedge P(u,v',w')) \supset v=v')$

holds on P, then a multivalued dependency

$\forall u \forall x \forall x' \forall w \forall w'((Q(u,x,w) \wedge Q(u,x',w')) \supset Q(u,x,w'))$

holds on Q. In this case, a projection transformation defined by

$Q(u,x,w) \rightarrow R(u,x)$

and

$Q(u,x,w) \rightarrow S(u,w)$

can be applied to Q and two relation schemata R and S are obtained. If x stands for a pair (y,z), then a deproduct transformation (a special encode transformation) defined by

$R(u,x) \rightarrow T(u,y,z)$

can be applied to R and a relation schema T is obtained. Finally, if a functional dependency

$\forall u \forall u' \forall y \forall z \forall z'((T(u,y,z) \wedge T(u',y,z')) \supset z=z')$

holds on T, then a projection transformation defined by

$T(u,y,z) \rightarrow U(u,y)$

and

$T(u,y,z) \rightarrow V(y,z)$

is again applicable to T. Eventually we have obtained three relation schemata S, U and V by a series of lossless schema transformations. It can be seen that a rule

$\forall u \forall y \exists w \exists z(U(u,y) \supset (S(u,w) \wedge V(y,z)))$

is generated after these schema transformations have been applied in a proper sequence. Figure 1 shows the

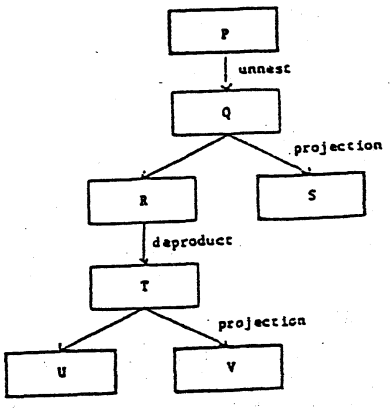

Figure 1: Generation of a relationship relation U.

whole transformation process. If a functional dependency

$$\forall u \forall w \forall w' ((S(u,w) \wedge S(u,w')) \supset w=w')$$

holds on $\underline{S}$, three relation schemata $\underline{S}$, $\underline{U}$ and $\underline{V}$ can be illustrated in the Entity-Relationship Diagram shown in Figure 2. In addition, an inclusion dependency

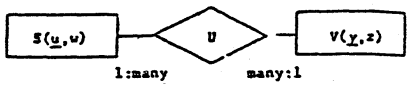

Figure 2: ER-Diagram of three relations S, U and V.

$$\forall y \forall z \exists u (V(y,z) \supset U(u,y))$$

holds. An inclusion dependency

$$\forall u \forall w \exists y (S(u,w) \supset U(u,y))$$

holds only when

$$\forall u \forall v \forall w (P(u,v,w) \supset v \neq \emptyset)$$

holds on $\underline{P}$.

In general a rule of the form

$$\forall x_1 \forall x_2 \ldots \forall x_n \forall y \exists z_1 \exists z_2 \ldots \exists z_n (R(x_1,x_2,\ldots,x_n,y) \supset U_1(x_1,z_1) \wedge U_2(x_2,z_2) \wedge \ldots \wedge U_n(x_n,z_n)))$$

is called a relationship rule provided that functional dependencies

$$\forall x_k \forall z_k \forall z_k' ((U_k(x_k,z_k) \wedge U_k(x_k,z_k')) \supset z_k = z_k')$$

holds on $\underline{U_1}, \underline{U_2}, \ldots, \underline{U_n}$. Here, $\underline{U_1}, \underline{U_2}, \ldots, \underline{U_n}$ are not necessarily mutually disjoint. A relationship rule may appear after repeatedly applied projection transformations to one or more relation schemata, or may be given a priori at the initial stage of database design. If such a rule holds, the relation schema $\underline{R}$ is called a relationship relation schema and a relation R in $\underline{R}$ is called a relationship relation. Tuples in a relationship relation are sometimes called relationships. The above gives a formal basis of notions regarding relationship relations, which have been discussed in a more intuitive and informal manner until today [CHE 1976].

3. CLASSIFICATION OF BINARY RELATIONSHIP RELATIONS

Let $\underline{R}$ be a binary relationship relation schema between two relation schemata $\underline{U}$ and $\underline{V}$, that is, functional dependencies

$$\forall x \forall u \forall u' ((U(x,u) \wedge U(x,u')) \supset u=u'),$$

and

$$\forall y \forall v \forall v' ((V(y,v) \wedge V(y,v')) \supset v=v'),$$

and a binary relationship rule

$$\forall x \forall y \exists u \exists v (R(x,y) \supset (U(x,u) \wedge V(y,v)))$$

hold over $\underline{R}$, $\underline{U}$ and $\underline{V}$. For simplicity, let us define that

$\Phi_1(x) \equiv \exists u U(x,u)$

and

$\Phi_2(y) \equiv \exists v V(y,v)$.

Then the relationship rule can be rewritten as

$\forall x \forall y (R(x,y) \supset (\Phi_1(x) \wedge \Phi_2(y)))$.

Such binary relationship relation schemata are particularly important because a binary relationship relation R in $\underline{R}$ can be regarded as a correspondence between U in $\underline{U}$ and V in $\underline{V}$, and also a directed graph whose nodes are tuples in UuV and arcs are tuples in R can be considered. Cardinality of the correspondence, and acyclicness and connectedness of the directed graph are important properties of the given binary relationship relation. Next let us classify binary relationship relation schemata with reference to these properties.

First let us describe four important properties of binary relationship relation schemata in the form of rules holding on them. The first two are functional dependencies

(FD) forward dependency: $\forall x \forall y \forall y' ((R(x,y) \wedge R(x,y')) \supset y=y')$,

and

(BD) backward dependency: $\forall x \forall x' \forall y ((R(x,y) \wedge R(x',y)) \supset x=x')$.

To describe the last two rules certain recursively defined predicate symbols become necessary, which were not necessarily introduced in defining algebraic and functional transformations [KOB 1986].

(AC) acyclicness: $\forall x \forall y (R(x,y) \supset \sim \Theta_R(y,x))$,

where $\Theta_R(x,y)$ is recursively defined by

$\Theta_R(x,y) \equiv R(x,y) \vee \exists z (\Theta_R(x,z) \wedge R(z,y))$.

(CN) connectedness: $\forall x \forall y ((\Phi_1(x) \wedge \Phi_2(y)) \supset \Gamma_R(x,y))$,

where $\Gamma_R(x,y)$ is recursively defined by

$\Gamma_R(x,y) \equiv R(x,y) \vee R(y,x) \vee \exists z (\Gamma_R(x,z) \wedge (R(z,y) \vee R(y,z)))$.

The forward dependency states that any relation R in $\underline{R}$ is a many-to-one correspondence from $\{x|\exists y R(x,y)\}$ to $\{y|\exists x R(x,y)\}$. As a directed graph it can not contain forward branches. The backward dependency states that any relation R in $\underline{R}$ is a one-to-many correspondence from $\{x|\exists y R(x,y)\}$ to $\{y|\exists x R(x,y)\}$. As a directed graph it cannot contain any backward branches. Sometimes a finer specification for the cardinality of correspondence can be made by giving a rule containing an aggregate function. For example, to specify that any element in $\{x|\exists y R(x,y)\}$ is corresponding to not less than μ and not more than ν

elements in $\{y|^{\exists}xR(x,y)\}$, a rule

$$^{\forall}x(\mu\leq(\Sigma;y;R(x,y))_{one}\leq\nu)$$

can be given, where $(\Sigma;y;R(x,y))_{one}$ is the aggregate function which summarizes constant 1 for all y satisfying R(x,y), that is, counts elements y satisfying R(x,y). If the both forward and backward dependencies hold, any relation R in $\underline{R}$ becomes a one-to-one correspondence. The third rule states that any relation R in $\underline{R}$ contains no circuits as a directed graph. The last rule states that it is connected as a directed graph.

Let $x\geq y$ be defined by

$$x\geq y\equiv x=y\vee\Theta_R(x,y).$$

The relation $\geq$ becomes a partial order on $\{x|\exists y(R(x,y)\vee R(y,x)\}$ if and only if the acyclicness holds. It becomes a full order if and only if all the forward and backward dependencies, acyclicness and connectedness hold.

Binary relationship relation schemata can be classified according to which rules holding on them. A binary relationship relation schema is called a pillar schema if all the forward dependency, backward dependency, acyclicness and connectedness hold on it. It is called a colonnade schema if the forward dependency, backward dependency and acyclicness hold on it. A binary relationship relation schema is called a tree schema if the backward dependency, acyclicness and connectedness hold on it. It is called a forest schema if the backward dependency and acyclicness hold on it. It is possible to define a backward tree schema and a backward forest schema by replacing the backward dependency with the forward dependency. All other binary relationship relation schemata are called network schemata. A network schema is said to be oriented if the acyclicness holds on it. It is said to be connected if the connectedness holds on it. There are several other combi-

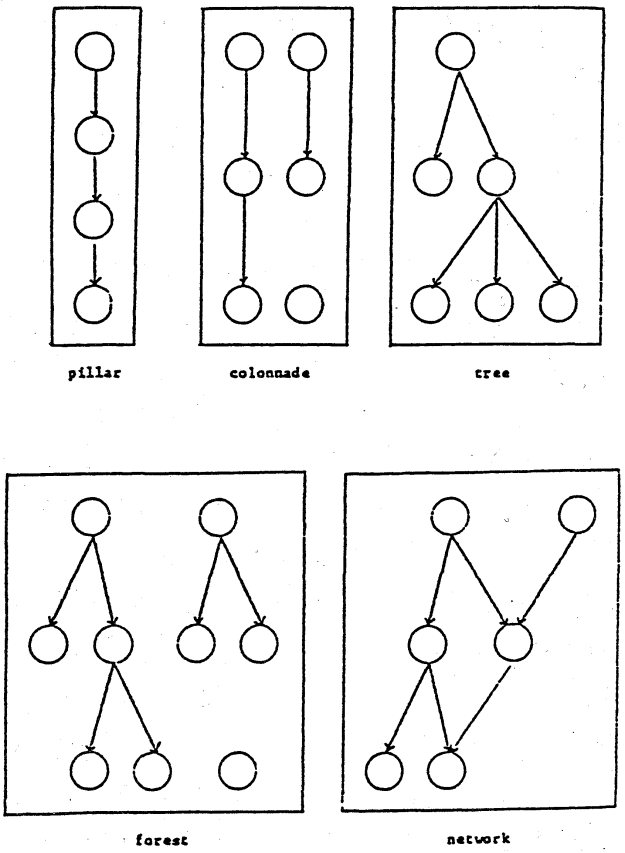

Figure 3: Various types of binary relationship relations.

nations of these four rules but only those mentioned above are important in database environment. A binary relationship relation is called a pillar, colonnade, tree, forest or network according to what type of schema it belongs. Figure 3 shows various types of binary relationship relations.

If for Φ_1 and Φ_2

$$\forall x(\sim\Phi_1(x)\vee\sim\Phi_2(x))$$

holds, that is,

$$\forall x\forall u\forall v(\sim U(x,u)\vee\sim V(x,v))$$

holds, then the relation schema $\underline{R}$ is said to be irrecursive. If $\underline{R}$ is irrecursive, any relation U in $\underline{U}$ and V in $\underline{V}$ do not intersect. Otherwise $\underline{R}$ is said to be recursive. For a recursive relationship relation schema, a rule

$$\forall x\forall u\forall v((U(x,u)\wedge V(x,v))\supset u=v)$$

is assumed in addition to functional dependencies

$$\forall x\forall u\forall u'((U(x,u)\wedge U(x,u'))\supset u=u')$$

and

$$\forall x\forall v\forall v'((V(x,v)\wedge V(x,v'))\supset v=v').$$

If an irrecursive forest schema $\underline{R}$ satisfies the rule

$$\forall y\exists x(\Phi_2(y)\supset R(x,y)),$$

then $\underline{R}$ is called a hierarchy schema with respect to a pair of predicate symbols (Φ_1,Φ_2). Relations in a hierarchy schema are called hierarchies. The last rule can be written as

$$\forall y\forall v\exists x(V(y,v)\supset R(x,y)),$$

which is an inclusion dependency. Figure 4 shows a hierarchy.

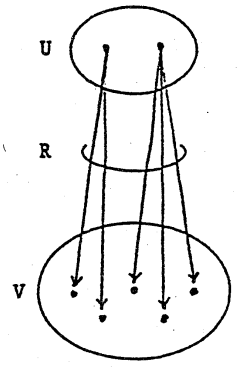

Figure 4: A Hierarchy.

The network model [BAC 1974] basically deals with hierarchies. A hierarchy is represented by an arrow in the Bachman Diagram. Shown in Figure 5 is a hierarchy R connecting U and V. In database management systems based on CODASYL DBTG proposal [CDS 1971] a hierarchy is defined by a SET command. Asserting the last inclusion dependency is optional in these systems.

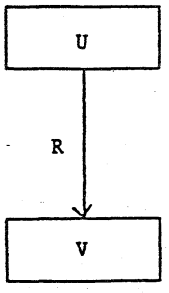

Figure 5: Bachman Diagram representing a hierarchy.

Let $\underline{R}_k$ (k=1,2,...,m) be hierarchy schemata each with respect to (Φ_{1k},Φ_{2k}). If rules

$$\forall x(\Phi_{11}(x)\supset(\sim\Phi_{22}(x)\wedge\sim\Phi_{23}(x)\wedge\ldots\wedge\sim\Phi_{2m}(x))),$$

$$\forall x(\Phi_{1k}(x)\supset(\Phi_{11}(x)\vee\Phi_{21}(x)\vee\Phi_{22}(x)\vee\ldots\vee\Phi_{2(k-1)}(x)))$$

for k=2,3,...,m, and

$$\forall x(\sim\Phi_{2i}(x)\vee\sim\Phi_{2j}(x))$$

for $i \neq j$, hold, then $\{\underline{R}_1, \underline{R}_2, \ldots, \underline{R}_m\}$ is called a hierarchically structured schema. A set of hierarchies in a hierarchically structured schema is called a hierarchical structure.

Let D_{ij} be the set $\{x|\Phi_{ij}(x)\}$. The first rules states that $D_{11} \cap D_{2k} = \emptyset$ $(k=2,3,\ldots,m)$. By the definition of hierarchies, D_{11} does not intersect D_{12}, either. The second rule states that for $k \geq 2$ at least one of $D_{1k} \subset D_{11}, D_{1k} \subset D_{21}, D_{1k} \subset D_{22}, \ldots, D_{1k} \subset D_{2(k-1)}$ holds. (Exactly one of them holds due to the third rule.) The third rule states that $D_{21}, D_{22}, \ldots, D_{2m}$ (and D_{11}) are mutually disjoint. Let $\{D'_1, D'_2, \ldots, D'_n\}$ be the set obtained from $\{D_{11}, D_{12}, \ldots, D_{1m}, D_{21}, D_{22}, \ldots, D_{2m}\}$ by eliminating D_{ij} if there is a set D_{pq} for which $D_{ij} \subset D_{pq}$. Let $\bar{R}(D'_\mu, D'_\nu)$ stands for a hierarchy schema $\underline{R}_{\mu\nu}$ whose origins are in D'_μ and destinations are in D'_ν exist. Then the above three rules can be interpreted as $\bar{R}$ becoming a tree. The relation corresponding to $\bar{R}$ is called the hierarchical skeleton of the given hierarchically structured schema. Figure 6 shows a hierarchical structure in a hierarchically structured schema and its skeleton.

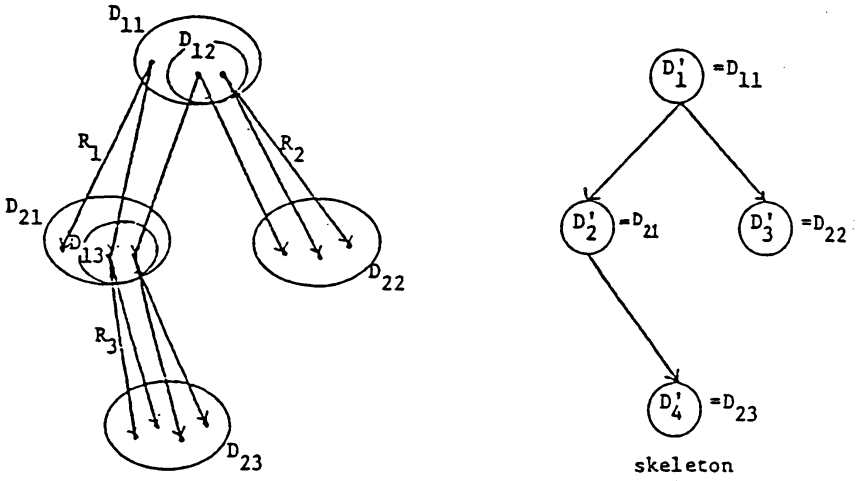

Figure 6: A hierarchical structure and its skeleton.

The hierarchical model [TSI 1976] deals with hierarchical structures. Since hierarchical structures very frequently appear in various business applications, certain special treatments of such structures were devised in most of the database management systems developed in early days.

4. GRAPH TRANSFORMATIONS

Let us next discuss four basic graph transformations each transforms a group of relationship relation schemata into another group of relationship relation schemata, and vice versa.

4.1. CONNECT/DISCONNECT TRANSFORMATION

Given two separate colonnades, it is sometimes possible to connect them

and generate a single colonnade, losslessly. This transformation is very simple but plays important roles in transforming complicated relationship relations into simples ones.

4.1.1. TRANSFORMATION RULES

Let $\underline{R}$ and $\underline{S}$ be corresponding to two-position predicate symbols R and S, respectively, and each represent a colonnade. Assume that two one-position predicate symbols Φ_1 and Φ_2 are given. The bidirectional transformation defined by

(G1F1) $R(x,y)\rightarrow T(x,y)$,

(G1F2) $S(x,y)\rightarrow T(x,y)$,

(G1B1) $T(x,y)\wedge\Phi_1(x)\rightarrow R(x,y)$,

and

(G1B2) $T(x,y)\wedge\Phi_2(x)\rightarrow S(x,y)$,

is called the connect/disconnect transformation or simply the G1-transformation.

Obviously, this is a special case of union/selection transformations (the reverse of selection/union transformation). However, it is of special importance when applied to colonnades.

4.1.2. RULES FOR THE TRANSFORMATION BEING BIJECTIVE

In the previous paper [KOB 1986], we have already seen rules necessary for a union/selection transformation being lossless. Two alternative sets of rules exist. In this special case, one is composed of

(G1O1) $\forall x(\sim\Phi_1(x)\vee\sim\Phi_2(x))$,

(G1I1) $\forall x\forall y(R(x,y)\supset\Phi_1(x))$,

(G1I2) $\forall x\forall y(S(x,y)\supset\Phi_2(x))$,

and

(G1II1) $\forall x\forall y(T(x,y)\supset(\Phi_1(x)\vee\Phi_2(x)))$.

The second set can be obtained by replacing the rule (G1O1) with two rules

(G1I3) $\forall x\forall y((R(x,y)\wedge\Phi_2(x))\supset S(x,y))$,

and

(G1I4) $\forall x\forall y((S(x,y)\wedge\Phi_1(x))\supset R(x,y))$.

We have assumed that relations in $\underline{R}$ and $\underline{S}$ are colonnades. Three more rules

(G1I5) $\forall x\forall x'\forall y((R(x,y)\wedge S(x',y))\supset x=x')$,

(G1I6) $\forall x\forall y(R(x,y)\supset\sim\Theta_{(R,S)}(y,x))$,

and

(GlI7) $\forall x \forall y (S(x,y) \supset \sim \Theta_{(R,S)}(y,x))$,

are necessary for relations in $\underline{T}$ being colonnades, where $\Theta_{(R,S)}(x,y)$ is recursively defined by

$$\Theta_{(R,S)}(x,y) \equiv R(x,y) \vee S(x,y) \vee \exists z (\Theta_{(R,S)}(x,z) \wedge (R(z,y) \vee S(z,y))).$$

Since $\underline{R}$ becomes acyclic if (GlI6) holds and $\underline{S}$ becomes acyclic if (GlI7) holds, the last two rules are a little stronger version of the acyclicness on a single schema.

To show that $\underline{T}$ becomes a colonnade schema if and only if $\underline{R}$ and $\underline{S}$ are colonnades and these three rules hold, it is sufficient to prove the forward dependency, backward dependency and acyclicness on $\underline{T}$ are rewritten into forward and backward dependencies on $\underline{R}$ and $\underline{S}$ plus (GlI5), (GlI6) and (GlI7) by the backward transformation. We can use several propositions applicable to transforming rules. The forward dependency

$$\forall x \forall y \forall y' ((T(x,y) \wedge T(x,y')) \supset y=y')$$

is transformed into

$$\forall x \forall y \forall y' (((R(x,y) \vee S(x,y)) \wedge (R(x,y') \vee S(x,y'))) \supset y=y'),$$

which is equivalent to

$$\forall x \forall y \forall y' ((R(x,y) \wedge R(x,y')) \supset y=y') \wedge \forall x \forall y \forall y' ((R(x,y) \wedge S(x,y')) \supset y=y')$$
$$\wedge \forall x \forall y \forall y' ((S(x,y) \wedge S(x,y')) \supset y=y').$$

If (Gl01) holds, R(x,y) and S(x,y') cannot hold at the same time. If (GlI3) holds, whenever R(x,y) and $\Phi_2(x)$ hold then S(x,y) holds. Since $\Phi_2(x)$ holds if S(x,y') holds, R(x,y') also holds in this case. In both cases, the second term is not necessary. Therefore, the first term, which is the forward dependency on $\underline{R}$, and the third term, which is the forward dependency on $\underline{S}$, remain.

A little different situation exist for the backward dependency

$$\forall x \forall x' ((T(x,y) \wedge T(x',y)) \supset x=x').$$

This is transformed into

$$\forall x \forall x' \forall y ((R(x,y) \wedge R(x',y)) \supset x=x')$$
$$\wedge \forall x \forall x' \forall y ((R(x,y) \wedge S(x',y)) \supset x=x')$$
$$\wedge \forall x \forall x' \forall y ((S(x,y) \wedge S(x',y)) \supset x=x').$$

In this case, all three terms are significant. The first term is the backward dependency on $\underline{R}$, the third term is the backward dependency on $\underline{S}$, and the second term is (GlI5). It is easy to see

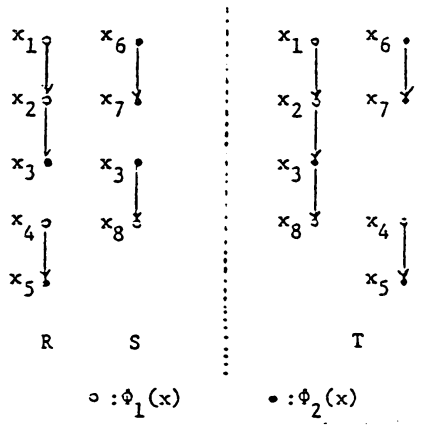

Figure 7: A connect/disconnect transformation.

that the acyclicness

$\forall x \forall y(T(x,y) \supset \sim\Theta_T(y,x))$

on $\underline{T}$ is rewritten into

$\forall x \forall y((R(x,y) \vee S(x,y)) \supset \sim\Theta_{(R,S)}(y,x))$

over $\underline{R}$ and $\underline{S}$, from which rules (G1I6) and (G1I7) can be obtained.

Figure 7 shows an example of G1-transformations.

4.2. TRIM/GRAFT TRANSFORMATION

In many cases, a forest is represented by a binary tree, or a hierarchical list. Devising such representation can be regarded as schema transformations from a pair composed of a forest schema and a colonnade schema into a pair of two colonnade schemata.

4.2.1. TRANSFORMATION RULES

Let $\underline{R}$ be a forest schema corresponding to a two-position predicate symbol R, and $\underline{S}$ and $\underline{T}$ be colonnade schemata corresponding to two-position predicate symbols S and T, respectively. The bidirectional transformation defined by

(G2F1) $R(x,y) \wedge \sim\exists z S(z,y) \rightarrow T(x,y)$,

(G2B1) $T(x,y) \rightarrow R(x,y)$,

and

(G2B2) $T(x,z) \wedge \Theta_S(z,y) \rightarrow R(x,y)$

is called the trim/graft transformation or simply the G2-transformation. The last transformation rule contains a recursively defined predicate symbol Θ_S. This rule can be rewritten as

(G2B2') $R(x,z) \wedge S(z,y) \rightarrow R(x,y)$

if the transformation rule is permitted to be applied recursively. Any relation S in $\underline{S}$ remains unchanged by the transformation.

4.2.2. RULES FOR THE TRANSFORMATION BEING BIJECTIVE

For the G2-transformation being lossless, several rules must hold over $\underline{R}$, $\underline{S}$ and $\underline{T}$ in addition to the backward dependency and acyclicness on $\underline{R}$, and the forward and backward dependencies and acyclicness on $\underline{S}$ and $\underline{T}$. Rules to hold over $\underline{R}$ and $\underline{S}$ are

(G2I1) $\forall x \forall y \forall z((R(x,y) \wedge R(x,z)) \supset \Gamma_S(y,z))$,

and

(G2I2) $\forall y \forall z \exists x(\Gamma_S(y,z) \supset (R(x,y) \wedge R(x,z)))$.

Rules to hold over $\underline{S}$ and $\underline{T}$ are

(G2II1) $\forall x\forall y\forall z(\sim S(x,z)\vee\sim T(y,z))$,

and

(G2II2) $\forall y\forall z\exists x(S(y,z)\supset(S(x,y)\vee T(x,y)))$.

A colonnade can be regarded as a collection of several pillars. The first two rules state that S in $\underline{S}$ is a collection of pillars each linking destinations of arcs in R in $\underline{R}$ with the common origin. The last two state that the origin of any arc in S in $\underline{S}$ should be the destination of an arc in S in $\underline{S}$ or T in $\underline{T}$ but not the both. It is easy to prove that the forward transformation becomes injective if and only if the first two rules hold, while it becomes surjective if and only if the last two rules hold. Figure 8 shows an example G2-transformation.

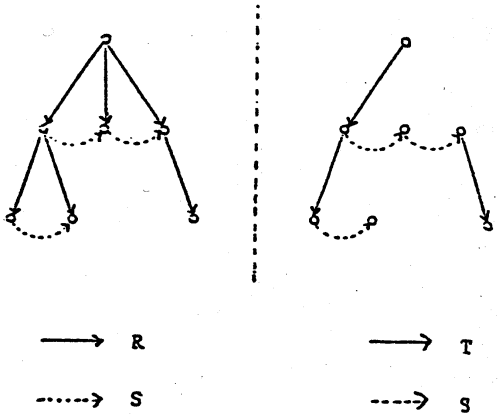

Figure 8: A trim/graft transformation.

4.2.3. COMBINATION OF A G1- AND A G2-TRANSFORMATION

Since a hierarchy schema $\underline{R}$ is a forest schema, the trim transformation defined by

$$R(x,y)\wedge\sim\exists z S(z,y)\rightarrow T(x,y)$$

can be applied if a colonnade schema $\underline{S}$ satisfying

$$\forall x\forall y\forall z((R(x,y)\wedge R(x,z))\supset\Gamma_S(y,z))$$

and

$$\forall y\forall z\exists x(\Gamma_S(y,z)\supset(R(x,y)\wedge R(x,z)))$$

is given. It can be seen that such a colonnade schema $\underline{S}$ can be defined whenever a hierarchy schema $\underline{R}$ is given. In this case, two predicate symbols Φ_1 and Φ_2, for which

$$\forall x(\sim\Phi_1(x)\vee\sim\Phi_2(x))$$

holds, are defined. Then two additional rules

$$\forall x\forall y(R(x,y)\supset(\Phi_1(x)\wedge\Phi_2(y))$$

and

$$\forall y\exists x(\Phi_2(y)\supset R(x,y))$$

holds on $\underline{R}$, and, in accordance, two additional rules

$$\forall x\forall y(S(x,y)\supset(\Phi_2(x)\wedge\Phi_2(y)))$$

and

$$\forall x\forall y(T(x,y)\supset(\Phi_1(x)\wedge\Phi_2(y)))$$

hold on S and T, respectively. In consequence, the connect transformation defined by

$S(x,y) \rightarrow U(x,y)$

and

$T(x,y) \rightarrow U(x,y)$

can be applied to obtain a single colonnade schema U. This schema satisfies all the forward dependency, backward dependency and acyclicness, and also two rules

$\forall x \forall y (U(x,y) \supset \Phi_2(y))$

and

$\forall x \forall y \exists z (U(x,y) \supset (U(z,x) \vee \Phi_1(x)))$.

Conversely when a colonnade schema U satisfies these rules, then applying the disconnect transformation defined by

$U(x,y) \wedge \Phi_1(x) \rightarrow T(x,y)$

and

$U(x,y) \wedge \Phi_2(x) \rightarrow S(x,y)$,

followed by the graft transformation defined by

$T(x,y) \rightarrow R(x,y)$

and

$T(x,z) \wedge \Theta_S(z,y) \rightarrow R(x,y)$,

the original hierarchy schema R is reconstructed.

The G2-transformation provides a means to convert a forest schema into a pair of colonnade schemata. A hierarchy schema is a special forest schema for which the above combination of G1- and G2-transformations provides a means to convert it into a single colonnade schema.

4.3. DISASSEMBLE/ASSEMBLE TRANSFORMATION

Any relationship relation schema can be decomposed into several hierarchy schemata. First let us show a schema transformation that transforms a binary relationship relation schema of any type (can be a network schema) into two hierarchy schemata.

4.3.1. TRANSFORMATION RULES

Let R be a binary relationship relation schema on which a relationship rule

$\forall x \forall y (R(x,y) \supset (\Phi_{11}(x) \wedge \Phi_{12}(y))$

holds. Assume that a two-position function symbol f is given. Let $\underline{S}$ and $\underline{T}$ be two hierarchy schemata with respect to (Φ_{11},Φ_2) and (Φ_{12},Φ_2), respectively.

The transformation defined by

(G3F1) $R(x,y)\rightarrow S(x,f(x,y))$,

(G3F2) $R(x,y)\rightarrow T(y,f(x,y))$

and

(G3B1) $S(x,z)\wedge T(y,z)\rightarrow R(x,y)$

is called the disassemble/assemble transformation or simply the G3-transformation.

The importance of this transformation is at the point that $\underline{R}$ can be a binary relationship relation schema of an arbitrary type, while both $\underline{S}$ and $\underline{T}$ are hierarchy schemata.

4.3.2. RULES FOR THE TRANSFORMATION BEING BIJECTIVE

For the G3-transformation being lossless, a rule

(G301) $\forall x\forall x'\forall y\forall y'((\Phi_{11}(x)\wedge\Phi_{11}(x')\wedge\Phi_{12}(y)\wedge\Phi_{12}(y')\wedge f(x,y)=f(x',y'))\supset(x=x'\wedge y=y'))$

must be satisfied. This rule states that the function f itself is an injection defined on

$$\{x|\Phi_{11}(x)\}\times\{y|\Phi_{12}(y)\}.$$

Any value standing for the pair (x,y) can be used as the function value. It will be obvious that the transformation is bijective if and only if above rule holds.

For the obtained schemata $\underline{S}$ and $\underline{T}$ being hierarchy schemata with respect to (Φ_{11},Φ_2) and (Φ_{12},Φ_2), respectively, two additional rules

(G302) $\forall x(\sim(\Phi_{11}(x)\vee\Phi_{12}(x))\vee\sim\Phi_2(x))$,

and

(G303) $\forall x\forall y((\Phi_{11}(x)\wedge\Phi_{12}(y))\supset\Phi_2(f(x,y)))$

must be satisfied. The rule (G302) is equivalent to

$$\forall x(\sim\Phi_{11}(x)\vee\sim\Phi_2(x))\wedge\forall x\forall y(\sim\Phi_{12}(x)\vee\sim\Phi_2(x)).$$

It will be easy to see that $\underline{S}$ and $\underline{T}$ become hierarchy schemata if and only if (G302) and (G303) hold.

There is no rules to hold on $\underline{R}$, except the relationship rule mentioned previously, that is, $\underline{R}$ can be a binary relationship relation schema of any type. On the other hand, $\underline{S}$ and $\underline{T}$ are hierarchy schemata. This means that relationship rules

$$\forall x\forall y(S(x,y)\supset(\Phi_{11}(x)\wedge\Phi_2(y)))$$

and

$\forall x \forall y(T(x,y) \supset (\Phi_{12}(x) \wedge \Phi_2(y)))$,

and inclusion dependencies

$\forall y \exists x(\Phi_2(y) \supset S(x,y))$

and

$\forall y \exists x(\Phi_2(y) \supset T(x,y))$

must hold. Figure 9 shows an example G3-transformation.

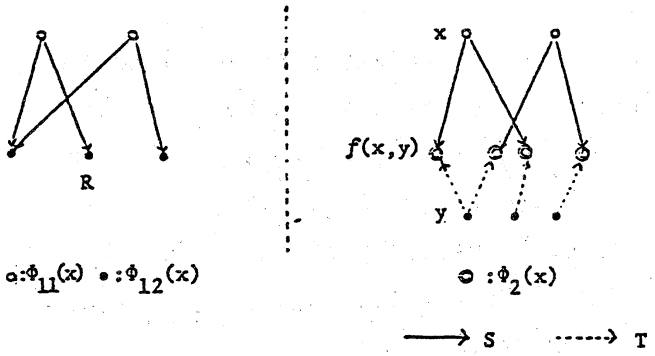

Figure 9: A disassemble/assemble transformation.

4.3.3. DISASSEMBLING IRRECURSIVE AND RECURSIVE RELATIONSHIP RELATION SCHEMATA

Assume that R is a relationship relation schema on which the relationship rule

$\forall x \forall y \forall z \exists u \exists v(R(x,y,z) \supset (U(x,u) \wedge V(y,v)))$

holds. Transformation rules can be generalized for dealing with such cases. Rules for the forward transformation are

(G3F1') $R(x,y,w) \rightarrow S(x,f(x,y))$,

(G3F2') $R(x,y,w) \rightarrow T(y,f(x,y))$

and

(G3F3') $R(x,y,w) \rightarrow W(f(x,y),w)$.

The backward transformation rule is

(G3B1') $S(x,z) \wedge T(y,z) \wedge W(z,w) \rightarrow R(x,y,w)$.

Now Φ_{11}, Φ_{12} and Φ_2 are defined as

$\Phi_{11}(x) \equiv \exists u U(x,u)$,

$\Phi_{12}(y) \equiv \exists v V(y,v)$

and

$\Phi_2(z) \equiv \exists w W(z,w)$.

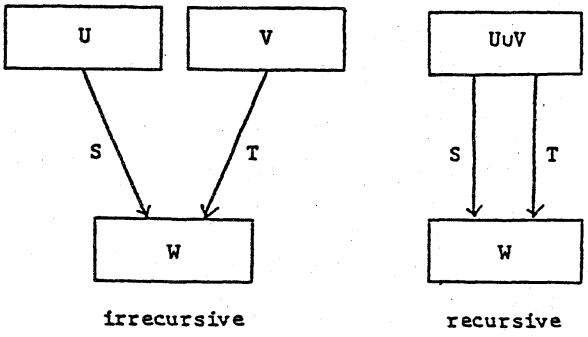

Figure 10: Hierarchies obtained from irrecursive and recursive relationship relations.

Then S can be regarded as a hierarchy schema defined between U and W, while T a hierarchy schema defined between V and W. If R is irrecursive, any relation U in U does not intersect any relation V in V. This is not the case if R is recursive. Let UuV be a relation schema whose relations are UuV for U in U and V in V. Then both S and T can be regarded as hierarchy schemata between UuV and W. Illustrated in Figure 10 are Bachman Diagrams corresponding to these two cases.

4.3.4 DISASSEMBLING N-ARY RELATIONSHIP RELATION SCHEMATA

The G3-transformation can be extended to that which disassembles an n-ary

relationship relation schema into n hierarchy schemata. Let $\underline{R}$ be an n-ary relationship relation schema on which a relationship rule

$$\forall x_1 \forall x_2 \ldots \forall x_n \forall y \exists z_1 \exists z_2 \ldots \exists z_n (R(x_1,x_2,\ldots,x_n,y) \supset (U_1(x_1,z_1) \wedge U_2(x_2,z_2) \wedge \ldots \wedge U_n(x_n,z_n)))$$

holds. Transformation rules for the extended G3-transformation are

(G3F1") $R(x_1,x_2,\ldots,x_n,y) \rightarrow S_k(x_k, f(x_1,x_2,\ldots,x_n))$
for k=1,2,...,n,

(G3F2") $R(x_1,x_2,\ldots,x_n,y) \rightarrow W(f(x_1,x_2,\ldots,x_n),y)$,

and

(G3B1") $S_1(x_1,z) \wedge S_2(x_2,z) \wedge \ldots \wedge S_n(x_n,z) \wedge W(z,y) \rightarrow R(x_1,x_2,\ldots,x_n,y)$.

Rules (G301), (G302) and (G303) should be extended to

(G301") $\forall x_1 \forall x_1' \forall x_2 \forall x_2' \ldots \forall x_n \forall x_n' ((\Phi_{11}(x_1) \wedge \Phi_{11}(x_1') \wedge \Phi_{12}(x_2) \wedge \Phi_{12}(x_2') \wedge \ldots \wedge \Phi_{1n}(x_n) \wedge \Phi_{1n}(x_n') \wedge f(x_1,x_2,\ldots,x_n) = f(x_1',x_2',\ldots,x_n')) \supset (x_1=x_1' \wedge x_2=x_2' \wedge \ldots \wedge x_n=x_n'))$,

(G302") $\forall x(\sim(\Phi_{11}(x) \vee \Phi_{12}(x) \vee \ldots \vee \Phi_{1n}(x)) \vee \sim\Phi_2(x))$

and

(G303") $\forall x_1 \forall x_2 \ldots \forall x_n ((\Phi_{11}(x_1) \wedge \Phi_{12}(x_2) \wedge \ldots \wedge \Phi_{1n}(x_n)) \supset \Phi_2(f(x_1,x_2,\ldots,x_n)))$,

respectively, where

$$\Phi_{1k}(x_k) \equiv \exists z_k U_k(x_k,z_k).$$

This transformation disassembles any n-ary relationship relation schema into n hierarchy schemata $\underline{S}_1,\underline{S}_2,\ldots,\underline{S}_n$ plus a relation schema $\underline{W}$.

Note that the disassembling is applicable even for n=1. This implies that if an inclusion dependency

$$\forall x \forall y \exists u (R(x,y) \supset U(x,u))$$

and a functional dependency

$$\forall x \forall u \forall u' ((U(x,u) \wedge U(x,u')) \supset u=u')$$

hold, then $\underline{R}$ can be decomposed into a hierarchy schema $\underline{S}$ and a relation schema $\underline{W}$ generated by rules

$$R(x,y) \rightarrow S(x,f(x)),$$

and

$$R(x,y) \rightarrow W(f(x),y).$$

The latter representation is a little more complicated than the former but $\underline{S}$ explicitly specifies the one-to-many correspondence from $\underline{U}$ onto $\underline{W}$.

4.4. INTEGRATE/DISINTEGRATE TRANSFORMATION

The connect/disconnect transformation connects two separate colonnades into a single colonnade, and vice versa. A little different transformation

that also generates a single colonnade from two separate colonnades and vice versa can be devised. Each of two colonnades to be combined is that obtained from a hierarchy and a colonnade by the trim transformation followed by the connect transformation.

4.4.1. TRANSFORMATION RULES

Assume that three one-position predicate symbols Φ_1, Φ_2 and Φ_3 are given. For three colonnade schemata R, S and T corresponding to two-position predicate symbols R, S and T, respectively, the transformation defined by

(G4F1) $R(x,y)\wedge\Phi_1(x)\rightarrow T(x,y)$,

(G4F2) $R(x,y)\wedge\sim\exists z S(x,z)\rightarrow T(x,y)$,

(G4F3) $S(x,y)\rightarrow T(x,y)$,

(G4F4) $R(w,y)\wedge\Theta_S(w,x)\wedge\sim\exists z S(x,z)\rightarrow T(x,y)$,

(G4B1) $T(x,y)\wedge\Phi_1(x)\rightarrow R(x,y)$,

(G4B2) $T(x,y)\wedge\Phi_2(x)\wedge\Phi_2(y)\rightarrow R(x,y)$,

(G4B3) $T(x,y)\wedge\Phi_3(y)\rightarrow S(x,y)$,

and

(G4B4) $T(w,y)\wedge\Phi_3(w)\wedge\Phi_2(y)\wedge\Theta_T(x,w)\wedge\Phi_2(x)$
$\wedge\sim\exists z(\Theta_T(x,z)\wedge\sim\Theta_T(w,z)\wedge\Phi_2(z))$
$\rightarrow R(x,y)$

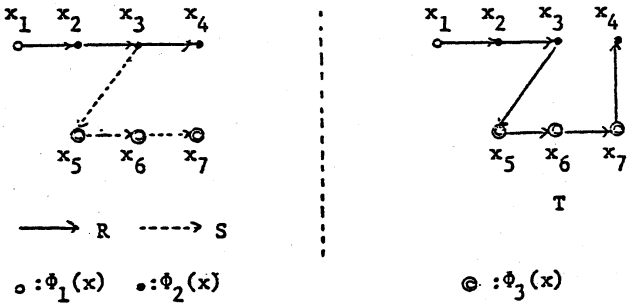

Figure 11: A integrate/disintegrate transformation.

is called the integrate/disintegrate transformation or simply the G4-transformation. The transformation rules are apparently very complicated; however, it can be understood by an example transformation shown in Figure 11.

4.4.2. RULES FOR THE TRANSFORMATION BEING BIJECTIVE

First, for Φ_1, Φ_2 and Φ_3, a rule

(G4O1) $\forall x((\sim\Phi_1(x)\vee\sim\Phi_2(x))\wedge(\sim\Phi_2(x)\vee\sim\Phi_3(x))\wedge(\sim\Phi_3(x)\vee\sim\Phi_1(x)))$

must hold. This implies three predicate symbols Φ_1, Φ_2 and Φ_3 are mutually exclusive.

On R and S, in addition to the forward and backward dependencies and acyclicness, rules

(G4I1) $\forall x\forall y\exists z(R(x,y)\supset(R(z,x)\vee\Phi_1(x))$,

(G4I2) $\forall x\forall y(R(x,y)\supset\Phi_2(y))$,

(G4I3) $\forall x\forall y\exists z(S(x,y)\supset(S(z,x)\vee\Phi_2(x)))$,

(G4I4) $\forall x\forall y(S(x,y)\supset\Phi_3(y))$

and

(G4I5) $\forall y \forall z \exists x((S(y,z) \wedge \Phi_2(y)) \supset R(x,y))$
must hold. On $\underline{T}$, in addition to forward and backward dependencies and acyclicness, rules
(G4II1) $\forall x \forall y \exists z(T(x,y) \supset (T(z,x) \vee \Phi_1(x)))$,
(G4II2) $\forall x \forall y((T(x,y) \wedge \Phi_1(x)) \supset \Phi_2(y))$
and
(G4II3) $\forall x \forall y(T(x,y) \supset (\Phi_2(y) \vee \Phi_3(y)))$
must hold. It will be easy to see that the transformation becomes bijective if and only if all these rules are satisfied.

4.4.3. CONVERTING A HIERARCHICALLY STRUCTURED SCHEMA INTO A COLONNADE SCHEMA

Let $\{\underline{R}_1, \underline{R}_2\}$ be a hierarchically structured schema, where $\underline{R}_1$ is a hierarchy schema with respect to (Φ_{11}, Φ_{21}) and $\underline{R}_2$ is that with respect to (Φ_{12}, Φ_{22}). According to the definition of hierarchically structured schemata, there can be two cases:
(CASE 1) $\forall x(\Phi_{12}(x) \supset \Phi_{11}(x))$ holds,
and
(CASE 2) $\forall x(\Phi_{12}(x) \supset \Phi_{21}(x))$ holds.
Obviously in the first case, $\underline{R}_1$ and $\underline{R}_2$ can be combined into a single hierarchy schema by defining

$$\Phi_1(x) \equiv \Phi_{11}(x)$$

and

$$\Phi_2(x) \equiv \Phi_{21}(x) \vee \Phi_{22}(x).$$

As already seen, this hierarchy schema can be transformed into a single colonnade schema. In the second case, $\underline{R}_1$ and $\underline{R}_2$ can be transformed into colonnade schemata $\underline{S}$ and $\underline{T}$, respectively. It is obvious that the above integrate transformation is applicable to these two colonnade schemata. In both cases, therefore, a hierarchically structured schema with two components can be transformed into a single colonnade schema losslessly. It will be easy to show that a hierarchically structured schema with more than two components can be transformed losslessly into a single colonnade schema by applying this procedure repeatedly. Note that this transformation can be applied to a forest schema to obtain a colonnade schema but it is, in general, a lossy transformation because the backward transformation is impossible.

5. BRIDGING GAPS AMONG DATA MODELS

Data models can be classified into two major categories. Data models in the first category do not make distinctions between entity and relationship relations. Information Algebra [CDS 1962], Relational Model [COD 1970], Extended Set Theory [CHI 1977] and Infological Model [SUN 1974] fall into this category. There can be various database schemata based on these models for representing very same world. Data translations among equivalent schemata can be described in terms of four basic transformations (algebraic and functional transformations) presented in the previous paper [KOB 1986].

Data models in the second category make distinctions between entity and relationship relations. Entity-Relationship Model [CHE 1976] deals with n-ary relationship relation schemata, while Information Space Model [KOB 1975a, 1975b] deals only with binary relationship relation schemata but of any type. Network Model [BAC 1974] can handle hierarchy schemata and some special pillar/colonnade schemata. More than two hierarchy schemata can be defined on a single relation schema. Hierarchical Model [TSI 1976] deals only with hierarchically structured schemata but only one hierarchically structured schemata can be defined on a single relation schema.

As mentioned in section 2, relationship relation schemata can be formally defined in terms of relationship rules. Binary relationship relation schemata are classified into several types according to which of the forward dependency, backward dependency, acyclicness and connectedness holding on them. These bridge the gap between data models in the first category and those in the second category.

Bridging the gap between Entity-Relationship Model (or Information Space Model) and Network Model, the disassembling/assembling transformation can be used. The definition of hierarchically structured schemata presented in section 3 may bridge the gap between Hierarchical Model and Network Model.

In existing database management systems, various statements are provided for defining relation schemata and rules holding on them. For example, in DBTG-type systems, a FILE statement defines a relation schema, while a SET statement defines a relation schema and also introduces two rules (relationship rule and backward dependency) to hold on it. Statements such as MANDATARY and AUTOMATIC assert certain rules to hold on binary relationship relation schemata. Making a precise correspondence between each statement and the defined schema/asserted rules may provide deeper and clearer understanding of data models and database management systems. It may also suggest a possibili-

ty of integrating deductive capability, possesed by PROLOG for example, into these database management systems.

Binary models such as Data Semantics [ABR 1974], DIAM [SEN 1973], Binary Logical Association [BRA 1976] and Extended Semantic Model [HAI 1975] are somewhat hybrid. A binary relation schema is sometimes dealt with as a flat relation, and sometimes as a correspondence. Anyway, data translations between these models and other models can be formally described using four basic schema transformations and four graph transformations discussed in this paper.

6. PHYSICAL REPRESENTATION OF BINARY RELATIONSHIP RELATIONS

The connect/disconnect transformation, trim/graft transformation and integrate/disintegrate transformation have a special importance in devising physical representation of binary relationship relations. This is based on the fact that a colonnade can be represented by a linear list.

Let $\underline{R}$ be a binary relationship relation schema on which a relationship rule

$$\forall x \forall y \exists u \exists v (R(x,y) \supset (U(x,u) \wedge V(y,v)))$$

holds. Here $\underline{R}$ can be either recursive or irrecursive. Let p be a function satisfying

$$\forall x \forall x' ((p(x) \wedge p(x')) \supset x = x').$$

This rule implies that p is an injection. A typical example of such functions is the memory address of the record representing the tuple (x,u). In this case, p is called a pointer.

The list representation of relations R in $\underline{R}$ can be regarded as the transformation defined by

(LF1) $U(x,u) \wedge \sim V(x,u) \wedge R(x,y) \rightarrow U'(x,u,p(y))$,

(LF2) $U(x,u) \wedge \sim \exists y R(x,y) \rightarrow U'(x,u,\perp)$,

(LF3) $V(x,u) \wedge R(x,y) \rightarrow V'(x,u,p(y))$,

(LF4) $V(x,u) \wedge \sim \exists y R(x,y) \rightarrow V'(x,u,\perp)$,

(LB1) $U'(x,u,z) \rightarrow U(x,u)$,

(LB2) $V'(x,u,z) \wedge z \neq \perp \rightarrow U(x,u)$,

(LB3) $V'(x,u,z) \rightarrow V(x,u)$,

(LB4) $U'(x,u,z) \wedge z \neq \perp \rightarrow R(x,p^{-1}(z))$,

(LB5) $V'(x,u,z) \wedge z \neq \perp \rightarrow R(x,p^{-1}(z))$.

Note that backward transformation rules assume the function p being a bijection. A rule

$\forall x\forall u\forall u'\forall z\forall z'(\sim U'(x,u,z)\vee\sim V'(x,u',z'))$

always hold, that is, any relation U' in $\underline{U}'$ does not intersect any relation V' in $\underline{V}'$.

Functional dependencies

$\forall x\forall u\forall u'\forall z\forall z'((U'(x,u,z)\wedge U'(x,u',z'))\supset(u=u'\wedge z=z'))$

and

$\forall x\forall u\forall u'\forall z\forall z'((V'(x,u,z)\wedge V'(x,u',z'))\supset(u=u'\wedge z=z'))$

hold if and only if $\underline{R}$ satisfies the forward dependency. Functional dependencies

$\forall x\forall x'\forall u\forall u'\forall z((U'(x,u,z)\wedge U'(x',u',z))\supset(x=x'\wedge u=u'))$,

and

$\forall x\forall x'\forall u\forall u'\forall z((V'(x,u,z)\wedge V'(x',u',z))\supset(x=x'\wedge u=u'))$,

and a rule

$\forall x\forall x'\forall u\forall u'\forall z(\sim U'(x,u,z)\vee\sim V'(x',u',z))$

hold if and only if $\underline{R}$ satisfies the backward dependency. Finally, two rules

$\forall x\forall u\forall z(U'(x,u,z)\supset\sim\Lambda(p^{-1}(z),x))$

and

$\forall x\forall u\forall z(V'(x,u,z)\supset\sim\Lambda(p^{-1}(z),x))$

hold, where $\Lambda(x,y)$ is recursively defined by

$\Lambda(x,y)\equiv\exists uV'(x,u,p(y))\vee\exists u\exists z(\Lambda(x,z)\wedge V'(z,u,p(y)))$,

if and only if $\underline{R}$ satisfies the acyclicness. The above seven rules together characterize the linear list representation. We have seen that the linear list representation is possible if and only if $\underline{R}$ is a colonnade schema.

The trim transformation provides a means of representing a forest by two pointers each comprising a linear list. If it is a hierarchy, the connect transformation can be applied after the trim transformation, and consequently it can be represented by a single pointer comprising a linear list. Futhermore, the integrate transformation provides a means of representing a hierarchical structure by a single pointer comprising a linear list.

It is possible to introduce additional transformations to append optional pointers to obtain a bidirectional list or a unidirectional or bidirectional ring. If the colonnade schema to be represented is that obtained by applying the trim transformation and connect transformation to a hierarchy schema $\underline{R}$, a backward pointer can be added by applying the list representation mentioned above for $\underline{R}'$ defined by

$R'(x,y)\equiv R(y,x)$.

These transformations can be considered as the foundation of physical repre-

sentation employed in existing database management systems based on Network and Hierarchical Models.

7. CONCLUSION

Data translations among various data models and hence various database management systems are important. Major difficulties in devising such data translations are caused by the reason that various notions are described in terminologies specific to data models. In particular, dealing with relationships and related notions is not a simple problem.

In this paper, relationships and several related notions have been formally described using rules (constraints) holding on relation schemata. Also four lossless schema transformations regarding binary relationship relation schemata have been presented. These may bridge gaps among various data models and database management systems.

List representation of binary relationship relations has been treated as a certain (lossless) schema transformation. This approach may suggest that some parts of the physical database organization problem can be dealt with as schema transformation problems.

REFERENCES

[ABR 1974] J.R.Abrial, Data Semantics, J.W.Klimbie, and K.L.Koffman, eds., Data Base Management, pp.1-59, North-Holland, Amsterdam, 1974.

[BAC 1974] C.W.Bachman, The Data Structure Set Model, *Proc. ACM SIGMOD '74*, pp.1-10, 1974.

[BRA 1976] G.Bracchi, P.Paolini, and G.Pelagatti, Binary Logical Associations in Data Modelling, G.M.Nijssen, ed., Modelling in Data Base Systems, pp.125-148, North-Holland, Amsterdam, 1976.

[CDS 1962] CODASYL Development Committee, An Information Algebra: Phase I report, *Comm. ACM*, Vol.5, No.4, pp.190-204, 1962.

[CDS 1971] CODASYL Data Base Task Group, April, 1971 report, ACM, 1971.

[CHE 1976] P.P.Chen, The Entity-Relationship Model: Toward a Unified View of Data, *ACM Trans. Database Syst.*, Vol.1, No.1, pp.9-36, 1976.

[CHI 1977] D.L.Childs, Extended Set Theory: A General Model for Very Large, Distributed, Backend Information Systems, *Proc. 3rd VLDB*, pp.28-46, 1977.

[COD 1970] E.F.Codd, A Relational Model of Data for Large Shared Data Banks, *Comm. ACM*, Vol.13, No.6, pp.377-387, 1970.

[HAI 1975] J.L.Hainaut, and B.Lacharlier, An Extensible Semantic Model of Data Base Systems and Its Data Language, *Proc. IFIP Congr.*, pp.1026-1030, North-Holland, Amsterdam, 1975.

[KOB 1975a] I.Kobayashi, Information and Information Processing Structure, *Int. J. Infor. Syst.*, Vol.1, No.2, pp.39-50, 1975.

[KOB 1975b] I.Kobayashi, DBTG Model, Relational Model and Information Space Model of the Information Structure, *Proc. 2nd USA-JAPAN Comp. Conf.*, pp.329-334, 1975.

[KOB 1986] I.Kobayashi, Losslessness and Semantic Correctness of Database Schema Transformation: Another Look of Schema Equivalence, To appear in *Int. J. Infor. Syst.*, Vol.11, No.1, 1986.

[SEN 1973] M.B.Senko, E.B.Altman, M.M.Astrahan, and P.L.Fehder, Data Structures and Accessing in Data-Base Systems, *IBM Syst. J.* Vol.12, No.1, pp.30-93, 1973.

[SUN 1974] B.Sundgren, Conceptual Foundation of the Infological Approach to Data Bases, J.W.Klimbie, and K.L.Koffman, eds., Data Base Management, pp.61-96, North-Holland, Amsterdam, 1974.

[TSI 1976] D.C.Tsichritzis, and F.H.Lochovsky, Hierarchical Data-Base Management: A Survey, *ACM Comp. Surv.*, Vol.8, No.2, pp.105-123, 1976.