DISCUSSION PAPER NO. 344

Transformations Between Relational Databases

by

Nancy D. Griffeth

Northwestern University
Graduate School of Management
Evanston, Illinois 60201

October, 1978

1. Introduction

An algorithm for transformations between two relational database systems is developed in this paper. There are three cases which must be considered:

(1) Transformations between equivalent database systems (equivalent in the sense that the class of derivable relations is the same in both cases);

(2) Transformations between comparable database systems (comparable in the sense that any relation derivable from one is derivable from the other); and

(3) Transformations between incomparable database systems.

In the case that database systems are equivalent, a canonical database for the equivalence class of database systems is used. The transformation from the source to the target database requires transformation from the source to the canonical database and then from the canonical database to the target database. In the case of comparable databases, the canonical forms for the databases may be different; therefore a transformation between canonical forms is also necessary. If the source "contains" the target, then the transformation is analogous to projection in Cartesian space. If the target "contains" the source, then some attributes must be assigned an undefined value in some relations.

The most difficult problem arises in the third case, when the source and the target database systems are incomparable. In this case, the source database cannot provide all of the relations that can be derived from the target database. Therefore, the real target of the transformation must be some database contained in the intended target. Furthermore, since there are relations derivable from the source database but not from the target database, the real target must also be properly contained in the source database. Information loss can therefore be minimized whenever the source and target databases have a greatest lower bound, using the partial order corresponding to the intuitive idea of containment used above. This follows because the greatest lower bound of the two databases is the "largest" database which is less than both the source and the target.

In this paper, the idea of "equivalence" of database systems is formalized and a canonical representation of an equivalence class is defined. Then the idea of containment is formalized as a partial order, and it is shown that the collection of equivalence classes of database systems together with the partial order corresponding to containment forms a lattice. Therefore, any pair of equivalence classes of database systems have a greatest lower bound. The greatest lower bound is the real target of a transformation between database systems. The transformation algorithm is summarized in figure 1. This approach is also developed in [5].

$$\begin{tikzcd}
(R_1,\ldots,R_k \arrow[r, "(1)"] & (S_1,\ldots,S_\ell) \arrow[dr, "(2)"'] & & (T_1,\ldots T_m) \arrow[r, "(4)"] & (U_1,\ldots,U_n) \\
& & \mathrm{glb}((S_1,\ldots,S_\ell),(T_1,\ldots,T_m)) \arrow[ur, "(3)"'] & &
\end{tikzcd}$$

Figure 1. The transformation algorithm.

In step (1), the database is transformed to the canonical representation for the equivalence class to which it belongs. In step (2), the canonical source database is projected onto the glb of the canonical source and canonical target databases. In step (3), the glb is transformed to the canonical target. In step (4), the canonical representation of the target database is transformed to the actual target database.

The paper is organized as follows:

Section 2. Framework. Relations, relational operators, and database systems are defined.

Section 3. A Database Transformation Problem. An example of the problems arising in database transformations is described.

Section 4. Equivalence Classes of Databases. The equivalence relation on databases is defined and the canonical representation for an equivalence class is given.

Section 5. The Lattice of Equivalence Classes. The existence of a greatest lower bound for each pair of equivalence classes of databases is proved. The transformation algorithm is given.

Section 6. Discussion. The construction of the lattice is discussed.

2. FRAMEWORK

The transformations studied take place in the relational model of databases, defined by Codd in [2]. A relation R is defined over a set $\alpha = \{A_1,\ldots,A_n\}$ of attributes. If the set of attributes must be specified to avoid ambiguity, we will write $R(\alpha)$ or $R(A_1,\ldots,A_n)$ (omitting the curly brackets in the latter case). Each attribute A_i is associated with a set of values $\mathrm{dom}(A_i)$. This set of values is called the domain of the attribute. The definition of domain is extended to sets of attributes as follows:

$$\mathrm{dom}(\alpha) = \{r \mid r \text{ is a function } \alpha \to \mathrm{dom}(A_1) \cup \ldots \cup \mathrm{dom}(A_n) \text{ such that } r(A_i) \in \mathrm{dom}(A_i)\}.$$

Each member of $\mathrm{dom}(\alpha)$ will be called a "tuple". For brevity, we will choose an ordering $\langle A_1,\ldots,A_n\rangle$ of α and write tuples r as $\langle r(A_1),\ldots,r(A_n)\rangle$. A relation $R(\alpha)$ is then a set of tuples belonging to $\mathrm{dom}(\alpha)$: $R(\alpha) \subseteq \mathrm{dom}(\alpha)$.

Two operations, projection and join, are used below. Projection of a tuple r belonging to $\mathrm{dom}(\alpha)$ onto a set β (written r_β) is the restriction of the domain of the function $r:\alpha \to \bigcup_{A\in\alpha} \mathrm{dom}(A)$ to the set $\alpha \cap \beta$. If $\alpha \cap \beta \neq \emptyset$, we say that $r_\beta = e$, the empty tuple. The definition of projection is extended to the relation $R(\alpha)$ by defining:

$$R(\alpha)_\beta = \{r_\beta \mid r \in R(\alpha)\} \quad \text{if} \quad \alpha \cap \beta \neq \emptyset;$$
$$R(\alpha)_\beta = \{e\} \quad \text{if} \quad \alpha \cap \beta = \emptyset .$$

The join $q*r$ of a tuple q belonging to $\mathrm{dom}(\beta)$ with a tuple r belonging to $\mathrm{dom}(\gamma)$ is defined:

$$\begin{aligned} q*r &= q \cup r \quad \text{if} \quad q(A) = r(A) \quad \text{for all} \quad A \in \beta \cap \gamma \\ &= e \quad \text{if} \quad q(A) \neq r(A) \quad \text{for all} \quad A \in \beta \cap \gamma \end{aligned}$$

The notation "$q \cup r$" should be understood to refer to the sets q and r defined as follows:

$$q = \{\langle A, q(A)\rangle | A \in \beta\}$$

$$r = \{\langle A, r(A)\rangle | A \in \gamma\}.$$

Thus $q*r \neq e$ if and only if the tuple $q \cup r$ is a function. Extend join to the relations $Q(\beta)$ and $R(\gamma)$ by defining:

$$Q(\beta) * R(\gamma) = \{q*r | q \in Q(\beta) \text{ and } r \in R(\gamma) \text{ and } q*r \neq e\}.$$

We say that the join of relations $Q(\beta)$ and $R(\gamma)$ is the natural join if

$$Q(\beta)_{\beta\cap\gamma} = R(\gamma)_{\beta\cap\gamma} .$$

In this case it can be shown that join and projection are inverses in the following sense:

$$Q(\beta) = (Q(\beta) * R(\gamma))_\beta$$

$$R(\gamma) = (Q(\beta) * R(\gamma))_\gamma .$$

A functional dependency $\beta \rightarrow \gamma$ between sets of attributes $\beta \subseteq \alpha$ and $\gamma \subseteq \alpha$ holds for a relation $R(\alpha)$ if $R(\alpha)_\beta \rightarrow R(\alpha)_\gamma$ is a function. It has been shown by Armstrong [1] that the family $\mathcal{F}$ of functional dependencies which hold for any relation $R(\alpha)$ obeys the following axioms:

(F1) For all $\beta \subseteq \alpha$, $\beta \rightarrow \beta \in \mathcal{F}$

(F2) If $\beta \rightarrow \gamma \in \mathcal{F}$ and $\gamma \rightarrow \delta \in \mathcal{F}$ then $\beta \rightarrow \delta \in \mathcal{F}$

(F3) If $\beta \rightarrow \gamma \in \mathcal{F}$ and, if $\beta \subseteq \beta'$ and $\gamma \supseteq \gamma'$, then $\beta' \rightarrow \gamma' \in \mathcal{F}$

(F4) If $\beta \rightarrow \gamma \in \mathcal{F}$ and $\delta \rightarrow \varphi \in \mathcal{F}$ then $\beta \cup \delta \rightarrow \gamma \cup \varphi \in \mathcal{F}$.

Armstrong defines any family $\mathcal{F}$ obeying these axioms as a full family of functional dependencies. Given any family $\mathcal{F}$ of functional dependencies, we will denote by $\mathcal{F}^*$ the smallest full family containing $\mathcal{F}$. Given any full family $\mathcal{F}$ of functional dependencies, we will denote by $\mathcal{R}(\mathcal{F})$ the set of relations $R(\alpha)$ for which all dependencies in $\mathcal{F}$ hold.

Finally, a <u>stored database</u> is a set of relations $R(\alpha(1)),\ldots,R(\alpha(n))$ such that $R(\alpha(i)) = R(\alpha)_{\alpha(i)}$ for some relation $R(\alpha)$. The set $\mathcal{A} = \{\alpha(1),\ldots,\alpha(n)\}$ is called a <u>database schema</u> if $\alpha(i) \subseteq \alpha(j)$ implies $\alpha(i) = \alpha(j)$. A <u>database system</u> is a triple $\Delta = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$ where α is a set of attributes, $\mathcal{F}$ is a full family of functional dependencies between subsets of α, and $\mathcal{A}$ is a database schema containing subsets of α. We assume the existence of a relation $R(\alpha)$ at each point in time such that the stored database for Δ is always equal to

$$\{R(\alpha)_{\alpha(1)},\ldots,R(\alpha)_{\alpha(n)}\} \text{ where } \mathcal{A} = \{\alpha(1),\ldots,\alpha(n)\}.$$

The relation $R(\alpha)$ need not be recoverable from the stored database, that is, it is not necessarily true that $R(\alpha) = R(\alpha)_{\alpha(1)} * \ldots * R(\alpha)_{\alpha(n)}$.

The following facts about joins and projections follow immediately from the definitions.

<u>Lemma 2.1</u>

If $\alpha \subseteq \beta$ and $\gamma \subseteq \delta$ and $\alpha \cup \gamma = \beta \cup \delta$ then

$$R_\alpha * R_\gamma \supseteq R_\beta * R_\delta$$

Proof:

Let $x \in R_\beta * R_\delta$, then for some $y, z \in R$, $x_\beta = y_\beta$, $x_\delta = z_\delta$, and $y_{\beta\cap\delta} = z_{\beta\cap\delta}$.

Therefore, $\alpha \subseteq \beta$ implies $x_\alpha = y_\alpha$; $\gamma \subseteq \delta$ implies $x_\gamma = z_\gamma$; and $\alpha \cap \gamma \subseteq \beta \cap \delta$ implies $y_{\alpha\cap\gamma} = z_{\alpha\cap\gamma}$. Then by definition of join:

$$x = y_\alpha * z_\gamma \in R_\alpha * R_\gamma$$

<u>Corollary 2.1</u>

$\alpha(i) \subseteq \beta(i)$ for $1 \leq i \leq n$ and $\cup\, \alpha(i) = \cup\, \beta(i)$ implies $*R_{\alpha(i)} \supseteq *R_{\beta(i)}$.

Proof follows from (1) by induction

<u>Lemma 2.2</u>

$R \subseteq R_\beta * R_\gamma$ for all β and γ such that $R \subseteq \mathrm{dom}(\beta \cup \gamma)$.

Proof:

Let $x \in R$, then $x = x_\beta * x_\gamma \in R_\beta * R_\gamma$.

<u>Corollary 2.2</u>

$R \subseteq R_{\alpha(1)} * \ldots * R_{\alpha(n)}$ for all $\alpha(1),\ldots,\alpha(n)$ such that $R \subseteq \mathrm{dom}(\alpha(1) \cup \ldots \cup \alpha(n))$.

Proof follows from (2) by induction.

<u>Lemma 2.3</u>

$(R_\beta * R_\gamma)_\delta \subseteq R_{\beta\cap\delta} * R_{\gamma\cap\delta}$ for all β, γ, δ.

Proof:

Let $x \in (R_\beta * R_\gamma)_\delta$. Then for some $y,z \in R, y_{\beta\cap\gamma}$ and $(y_\beta * z_\gamma)_\delta = x$.

$y_{\beta\cap\gamma} = z_{\beta\cap\gamma}$ implies $y_{\beta\cap\gamma\cap\delta} = z_{\beta\cap\gamma\cap\delta}$. $x = (y_\beta * z_\gamma)_\delta$ implies $x_\beta = y_{\beta\cap\gamma}$ and $x_\gamma = z_{\gamma\cap\delta}$. Therefore $x = y_{\beta\cap\delta} * z_{\gamma\cap\delta}$. ∎

<u>Corollary 2.3</u>

$(R_{\alpha(1)} * \ldots * R_{\alpha(n)})_\delta \subseteq R_{\alpha(1)\cap\delta} * \ldots * R_{\alpha(n)\cap\delta}$.

Proof follows from (3) by induction.

<u>Lemma 2.4</u>

If $\delta \subseteq \alpha \cup \beta$, then $(R_\alpha * R_\beta * R_\gamma)_\delta \subseteq (R_\alpha * R_\beta)_\delta$.

Proof:

Let $w \in (R_\alpha * R_\beta * R_\gamma)_\delta$. For some $x,y,z \in R, x_{\alpha\cap\beta} = y_{\alpha\cap\beta}$, $x_{\alpha\cap\gamma} = y_{\alpha\cap\gamma}$,

$y_{\beta\cap\gamma} = z_{\beta\cap\gamma}$, $w_\alpha = x_{\alpha\cap\delta}$, $w_\beta = y_{\beta\cap\delta}$ and $w_\gamma = z_{\gamma\cap\delta}$.

$\delta \subseteq \alpha \cup \beta$ implies $w = w_\delta = w_\alpha * w_\beta = x_{\alpha\cap\delta} * y_{\beta\cap\delta} = (x_\alpha * y_\beta)_\delta \in (R_\alpha * R_\beta)_\delta$. ∎

<u>Corollary 2.4</u>

If $\delta \subseteq \alpha(i) \cup \alpha(j)$ then $(R_{\alpha(1)} * \ldots * R_{\alpha(n)})_\delta \subseteq (R_{\alpha(i)} * R_{\alpha(j)})_\delta$.

Proof follows from (4) by induction.

Lemma 2.5

$$\text{If } R_\delta = (R_\alpha * R_\beta)_\delta \text{ then } R_\delta = (R_\alpha * R_\beta * R_\gamma)_\delta.$$

Proof:

If $R_\delta = (R_\alpha * R_\beta)_\delta$ then $\delta \subseteq \alpha \cup \beta$. By (4) $(R_\alpha * R_\beta * R_\gamma)_\delta \subseteq (R_\alpha * R_\beta)_\delta$. By (2a), $R_{\alpha\cup\beta\cup\gamma} \subseteq R_\alpha * R_\beta * R_\gamma$ and therefore $(R_\alpha * R_\beta)_\delta = R_\delta =$ $(R_{\alpha\cup\beta\gamma})_\delta \subseteq (R_\alpha * R_\beta * R_\gamma)_\delta$.
$\blacksquare$

3. A DATABASE TRANSFORMATION PROBLEM

A hypothetical firm handles ongoing administrative functions required for maintaining the organization through permanent departments, but all other work in the firm is organized into short-term projects which terminate when their goals are accomplished. A database system is used to keep track of current projects. The following relation describes the information stored in the database:

$R(\alpha)$ where $\alpha = \{E,P,L,D\}$ and the attributes are defined as follows:

E - employee identifier	$\text{dom}(E) = Z^+$
P - project title	$\text{dom}(P) = (\Sigma \cup \{\not{b}\})^+$
L - project leader's employee identifier	$\text{dom}(L) = Z^+$
D - department	$\text{dom}(D) = (\Sigma \cup \{b\})^+$

$(\Sigma = \{a,b,c,\ldots\} \cup \{A,B,C,\ldots\}$ and $Z^+ = \{1,2,3,\ldots\})$

A tuple $\langle e,p,\ell,d\rangle$ belongs to R whenever an employee e reports to project leader ℓ for project p. Project leader ℓ and employee e both work in department d.

Functional dependencies can be deduced from the following observations:

(1) $E \to D$, $L \to D$: An employee works in only one department.

(2) $DP \to L$: Projects may involve workers from several departments. Departmental structure is maintained within projects so that there is a single project leader in each department involved in the project.

(3) $EP \to L$: Each employee reports to the project leader in his own department, for each project he works on.

Two reports are produced from the database:

(1) The project report: Projects are listed by involved department.

(2) The project control report: All employees are listed by project, and under project by project leader.

Many collections of relations could be used to store the described information and produce the reports. Two such collections of relations are:

$$R_1(E,D),\ R_2(E,P),\ R_3(D,P,L)$$

and

$$S_1(P,D),\ S_2(P,E,L)$$

The formal definitions of these collections of relations as database systems is given in Figure 1a. A relation R(E,D,P,L) is illustrated in Figure 1b and the corresponding stored databases are given in Figures 1c and 1d. Both collections R_1,R_2,R_3 and S_1,S_2, are in the third normal form. The second collection has the advantage that the reports can be printed directly. The first collection has the advantage that the original relation R(E,P,L,D) can be recovered.

The transformation problem arises because generation of the reports requires generation of relations S_1 and S_2. If the database consists of relations R_1, R_2, and R_3 then these must be transformed to relations S_1 and S_2 to generate the reports. Conversely, if the database consists of relations S_1 and S_2, and if any new reports are to be produced, which are not part of one of those relations, then a transformation must also be performed.

First, we will consider the transformation from $\{R_1,R_2,R_3\}$ to $\{S_1,S_2\}$. Any of the following transformations might be tried:

(1a) $S_1(P,D) = R_3(D,P,L)_{PD}$

(1b) $S_1(P,D) = (R_1(E,D)*R_2(E,P))_{PD}$

(2a) $S_2(P,E,L) = (R_1(E,D)*R_3(D,P,L))_{PEL}$

(2b) $S_2(P,E,L) = (R_2(E,P)*R_3(D,P,L))_{PEL}$

(2c) $S_2(P,E,L) = (R_1(E,D)*R_2(E,P)*R_3(D,P,L))_{PEL}$

The transformation $T_1:\{R_1,R_2,R_3\} \rightarrow S_1$ and $T_2:\{R_1,R_2,R_3\} \rightarrow S_2$ must have the property that for every relation $R \in \mathcal{R}(\mathcal{F})$, $T_i(\{R_1,R_2,R_3\}) = R_{\beta(i)}$ where $\beta(1) = \{P,D\}$ and $\beta(2) = \{P,W,L\}$. The following theorem (proved by Heath [6], and Delobel [3], Rissanen [7]) tells us which transformations may be used:

<u>Theorem 3.1</u>. If $\beta,\gamma \subseteq \alpha$ then

$$(R(\alpha))_{\beta\cap\gamma} = R(\alpha)_\beta * R(\alpha)_\gamma \quad \text{for all } R \in \mathcal{R}(\mathcal{F})$$

if and only if

$$\beta \cap \gamma \rightarrow \beta \quad \text{or} \quad \beta \cap \gamma \rightarrow \gamma.$$

By the theorem, transformations (1a),(1b), and (2c) are valid; transformations (2a) and (2b) are not. (This is illustrated in Figure 1e). Conversely, suppose that that stored relations are S_1 and S_2 and the reports to be generated are represented by R_1, R_2, and R_3. Then we may try the following transformations:

(1) $R_1 = (S_1*S_2)_{ED}$ (2) $R_2 = (S_2)_{EP}$ (3) $R_3 = (S_1*S_2)_{DPL}$

By the theorem, transformation (2) is valid; transformation (1) and (3) are not. Thus relations R_1 and R_3 cannot be recovered at all from S_1 and S_2.

In subsequent sections of the paper, the following questions are answered:

(1) Given two collections of relations, when (in general) can we recover one collection from the other?

(2) Given that one collection of relations can be recovered from the other, what transformation should be used?

$\alpha = \{E, P.L, D\}$

$\mathcal{F} = \{E \rightarrow D, L \rightarrow D, PD \rightarrow L\}^*$

$\mathcal{A} = \{ED, EP, DPL\}$, $\mathcal{B} = \{PD, PEL\}$

$\Delta_1 = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$, $\Delta_2 = \langle \alpha, \mathcal{F}, \mathcal{B} \rangle$

(a)

R:

E	D	P	L
Adams	Accounting	Design	Adams
Boone	Accounting	Design	Adams
Boone	Accounting	Retool	Boone
Boone	Accounting	Study	Boone
Cowley	Accounting	Study	Boone
Davis	Production	Design	Davis
Elgin	Production	Retool	Elgin
Ford	Production	Retool	Elgin
Ford	Production	Study	Ford

(b)

Stored database Δ_1

R_1:

E	D
Adams	Accounting
Boone	Accounting
Cowley	Accounting
Davis	Production
Elgin	Production
Ford	Production

R_2:

E	P
Adams	Design
Boone	Design
Davis	Design
Boone	Retool
Elgin	Retool
Ford	Retool
Boone	Study
Cowley	Study
Ford	Study

R_3:

D	P	L
Accounting	Design	Adams
Accounting	Retool	Boone
Accounting	Study	Boone
Production	Design	Davis
Production	Retool	Elgin
Production	Study	Ford

(c)

Stored database Δ_2

S_1:

D	P
Accounting	Design
Accounting	Retool
Accounting	Study
Production	Design
Production	Retool
Production	Study

S_2:

P	E	L
Design	Adams	Adams
Design	Boone	Adams
Retool	Boone	Boone
Study	Boone	Boone
Study	Cowley	Boone
Design	Davis	Davis
Retool	Elgin	Elgin
Retool	Ford	Elgin
Study	Ford	Ford

(d)

$R_1(E,D) * R_3(D,P,L)$

E	D	P	L
Adams	Accounting	Design	Adams
Adams	Accounting	Retool	Boone
Adams	Accounting	Study	Boone
Boone	Accounting	Design	Adams
Boone	Accounting	Retool	Boone
Boone	Accounting	Study	Boone
Cowley	Accounting	Design	Adams
Cowley	Accounting	Retool	Boone
Cowley	Accounting	Study	Boone
Davis	Production	Design	Davis
Davis	Production	Retool	Elgin
Davis	Production	Study	Ford
Elgin	Production	Design	Davis
Elgin	Production	Retool	Elgin
Elgin	Production	Study	Ford
Ford	Production	Design	Davis
Ford	Production	Retool	Elgin
Ford	Production	Study	Ford

(e)

Figure 2. Two databases over relation R(E,P,D,D)

(a) Databases Δ_1 and Δ_2 are defined

(b) Relation R(E,P,L,D)

(c) A stored database Δ_1

(d) A stored database Δ_2

(e) An invalid transformation:

$R(E,P,L,D) \neq R_1(E,D) * R_3(D,P,L)$

4. EQUIVALENCE CLASSES OF DATABASES

A characterization of the information content of a database is developed in this section, from which it can be immediately determined whether one database can be derived from another. An equivalence relation for databases is then defined, such that two databases are said to be equivalent if each can be derived from the other. Finally, the canonical representation of an equivalence class is defined.

Before developing the formal characterization of the information content of a database, let's consider how we decided that Δ_2 can be derived from Δ_1 but not vice-versa. To show that Δ_2 can be derived from Δ_1, we displayed, for each relation in Δ_2, a transformation on the relations in Δ_1 whose value is the relation in Δ_2. That is:

$$S_1 = T_1(R_1,R_2,R_3) = (R_1{*}R_2)_{PD}$$

$$S_2 = T_2(R_1,R_2,R_3) = (R_1{*}R_2{*}R_3)_{PEL}$$

Each transformation consists of a series of joins followed by a projection; in fact, the following lemma shows that any relation derivable by a combination of natural joins and projections will be equal to a series of natural joins followed by a single projection.

Lemma 4.1

If $R(\alpha)_\delta = (R(\alpha)_\beta)_{\delta_1} * (R(\alpha)_\gamma)_{\delta_2}$
then
$$R(\alpha)_\delta = (R(\alpha)_\beta * R(\alpha)_\gamma)_\delta .$$

Proof:

$$R(\alpha)_\delta = (R(\alpha)_\beta)_{\delta_1} * (R(\alpha)_\gamma)_{\delta_2}$$

$$\supseteq (R(\alpha)_\beta)_\delta * (R(\alpha)_\gamma)_\delta \quad \text{(by Lemma 2.1)}$$

$$\supseteq (R(\alpha)_\beta * R(\alpha)_\gamma)_\delta . \quad \text{(by Lemma 2.3)}$$

But $R(\alpha) \subseteq R(\alpha)_\beta * R(\alpha)_\gamma$ (by Lemma 2.2)

implies $R(\alpha)_\delta \subseteq (R(\alpha)_\beta * R(\alpha)_\gamma)_\delta$

Therefore $R(\alpha)_\delta = (R(\alpha)_\beta * R(\alpha)_\gamma)_\delta$. ∎

By the lemma, we can always postpone projection to the last operation.

Thus, we will consider a relation $S(\beta)$ to be <u>derivable</u> from relations $R(\alpha(1)),\dots,R(\alpha(n))$ iff $\beta \subseteq \bigcup_{j \in J} \alpha(j)$ for some $J \subseteq \{1,\dots,n\}$ such that $R(\alpha)_\beta = (\mathop{*}_{j \in J} R(\alpha(j)))_\beta$. Then if we take a list of maximal sets β we need only look through the list to determine if $S(\beta)$ is derivable from $R(\alpha(1)),\dots,R(\alpha(n))$. Since for $\gamma \subseteq \beta$, $R(\alpha)_\gamma = ((R(\alpha))_\beta)_\gamma$, sets $\gamma \subseteq \beta$ for β in the list will be derivable.

4.1. Information Constraint Sets

The <u>information constraint set</u> of a database characterizes its information content. It is the maximal database schema of the databases in an equivalence class, in the sense of the partial order $\textcircled{\le}$ defined as follows:

<u>Definition</u>

For database schemas $\mathcal{A}$ and $\mathcal{B}$, we say that $\mathcal{A} \textcircled{\le} \mathcal{B}$ if for each $\alpha \in \mathcal{A}$, there exists some $\beta \in \mathcal{B}$ such that $\alpha \subseteq \beta$.

A partial order is reflexive, transitive, and antisymmetric. It follows immediately that $\textcircled{\le}$ is a partial order from the same properties of $\subseteq$ and from the fact that for any database schema $\mathcal{A}$ if $\alpha,\gamma \in \mathcal{A}$ and $\alpha \subseteq \gamma$ then $\alpha = \gamma$. (This last condition is necessary for antisymmetry.)

The following notation will be used to simplify subsequent definitions:

Let $\mathcal{B}$ be a database schema over a set α of attributes and let $\gamma \subseteq \alpha$. Then $\mathcal{B}(\gamma) = \{\beta | \beta \in \mathcal{B}$ and $\beta \cap \gamma \neq \emptyset\}$.

And the join of all relations R satisfying property P(R) will be written: $\mathop{*}_{P(R)} R$.

The information constraint set of a database Δ is defined so that, for any set of attributes γ contained in the information constraint set, the relation R_γ can be derived from the stored database; and any relation R_δ which can also be so derived has $\delta \subseteq \gamma$ for some γ in the information constraint set.

<u>Definition</u>

Given a database system $\Delta = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$ the <u>information constraint set</u> of Δ (written ics(Δ)) is the database schema satisfying the following conditions:

(1) for all $\gamma \in \text{ics}(\Delta)$, if $\gamma \subseteq \beta$ for some $\beta \in \text{ics}(\Delta)$ then $\gamma = \beta$.

(2) If $\gamma \in \text{ics}(\Delta)$ then $R_\gamma = (\mathop{*}\limits_{\beta \in \mathcal{A}(\gamma)} R_\beta)_\gamma$

(3) $R_\gamma = (\mathop{*}\limits_{\beta \in \mathcal{A}(\gamma)} R_\beta)_\gamma$ for arbitrary $\gamma \subseteq \alpha$ if and only if $\gamma \subseteq \gamma'$ for some $\gamma' \in \text{ics}(\Delta)$.

The following theorem shows that information constraint sets are well-defined.

<u>Theorem 4.1</u>. There is exactly one database schema satisfying the definition of an information constraint set for a given database.

Proof:

First, we show that there is at most one such schema. Suppose that we have a database $\Delta = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$ and two database schemas $\mathcal{C}$ and $\mathcal{B}$ satisfying the conditions. Then for each $\gamma \in \mathcal{C}$ we have (by condition (2)) that

$$R_\gamma = (\mathop{*}\limits_{\beta \in \mathcal{A}(\gamma)} R_\beta)_\gamma$$

and thus (by condition (3)) there is some $\delta \in \mathcal{B}$ such that $\delta \subseteq \gamma$. Therefore $\mathcal{C} = \mathcal{B}$. Finally, we show that there is at least one such schema. Let $\mathcal{C}$ be any minimal element of $\{\mathcal{B} | \mathcal{A} \leq \mathcal{B}$ and $\mathcal{B}$ satisfies condition 3$\}$. Then $\mathcal{C}$ must satisfy condition (2). This follows by contradiction: If $\mathcal{C}$ does not satisfy condition (2), then for some $\gamma \in \mathcal{C}$,

$$R_\gamma \neq (\mathop{*}\limits_{\beta \in \mathcal{A}(\gamma)} R_\beta)_\gamma \ .$$

Let $\gamma_1,\dots,\gamma_n$ be the maximal subsets of γ such that

$$R_{\gamma_i} = \Big(\mathop{*}_{\beta\in\mathcal{A}(\gamma_i)} R_\beta\Big)_{\gamma_i}$$

Then $\mathcal{C}' = \mathcal{C} - \{\gamma\} \cup \{\gamma_1,\dots,\gamma_n\}$ still satisfies condition (3) by definition of the γ_i ($\gamma_i \subseteq \gamma$ for each i). Since $\mathcal{C}' \le \mathcal{C}$, $\mathcal{C}$ could not have been minimal. $\blacksquare$

The following properties of an information constraint set follow immediately from the definition:

Lemma 4.1

If $\Delta = \langle \alpha,\mathcal{F},\mathcal{A}\rangle$ then $\mathcal{A} \textcircled{$\le$} \mathrm{ics}(\Delta)$

Proof: Substitute $\beta \in \mathcal{A}$ for γ in condition (3).

Lemma 4.2

If $\Delta_1 = \langle \alpha,\mathcal{F},\mathcal{B}\rangle$ and $\Delta_2 = \langle \alpha,\mathcal{F},\mathcal{C}\rangle$ and if $\mathcal{B} \textcircled{$\le$} \mathrm{ics}(\Delta_2)$ then $\mathrm{ics}(\Delta_1) \textcircled{$\le$} \mathrm{ics}(\Delta_2)$.

Proof:

Take

$\gamma \in \mathrm{ics}(\Delta_1)$.

$$R_\gamma = \Big(\mathop{*}_{\beta\in\mathcal{B}(\gamma)} R_\beta\Big)_\gamma \qquad \text{(by condition (2) for } \mathrm{ics}(\Delta_1))$$

$$= \Big(\mathop{*}_{\substack{\beta'\in \mathrm{ics}(\Delta_2)\\ \beta'\supseteq\beta\in\mathcal{B}(\gamma)}} R_{\beta'}\Big)_\gamma \qquad \text{(by Lemma 2.1)}$$

$$= \Big(\mathop{*}_{\substack{\beta'\in \mathrm{ics}(\Delta_2)\\ \beta'\supseteq\beta\in\mathcal{B}(\gamma)}} \Big(\mathop{*}_{\delta\in\mathcal{C}(\beta')} R_\delta\Big)_{\beta'}\Big)_\gamma \qquad \text{(by condition (2) for } \mathrm{ics}(\Delta_2))$$

$$= \Big(\mathop{*}_{\substack{\beta'\in \mathrm{ics}(\Delta_2)\\ \beta'\supseteq\beta\in\mathcal{B}(\gamma)}} \mathop{*}_{\delta\in\mathcal{C}(\beta')} R_\delta\Big)_{\gamma\cap(\cup\beta')} \qquad \text{(by Lemma 2.)}$$

$$= \Big(\mathop{*}_{\delta\in\mathcal{C}(\gamma)} R_\delta * \Big(\mathop{*}_{\delta\in\mathcal{C}(\cup\beta')-\mathcal{C}(\gamma)} R_\delta\Big)\Big)_\gamma$$

($\delta \cap \beta' \neq \emptyset$ for $\beta' \in \mathrm{ics}(\Delta_2)$) and $\beta' \supseteq \beta \in \mathcal{B}(\gamma)$; therefore $\beta' \cap \gamma \neq \emptyset$. Either δ intersects γ nontrivially or it intersects some β and not γ.)

$$= \Big(\mathop{*}_{\delta\in\mathcal{C}(\gamma)} R_\delta\Big)_\gamma. \qquad \text{(by Lemma 2.)}$$

Therefore, $\gamma \in \mathrm{ics}(\Delta_2)$. $\blacksquare$

Lemma 4.3

Database $\Delta_1 = \langle \alpha, \mathcal{F}, \mathcal{B} \rangle$ is contained in database $\Delta_2 = \langle \alpha, \mathcal{F}, \mathcal{C} \rangle$ in the following sense:

$$R_\beta = (\mathop{*}_{\gamma \in \mathcal{C}(\beta)} R_\gamma)_\beta \qquad \text{for all } \beta \in \mathcal{B}$$

if and only if $\mathrm{ics}(\Delta_1) \textcircled{$\leq$} \mathrm{ics}(\Delta_2)$.

Proof:

By condition (3), $\mathcal{B} \textcircled{$\leq$} \mathrm{ics}(\Delta_2)$ and by Lemma 4.2 above, $\mathrm{ics}(\Delta_1) \textcircled{$\leq$} \mathrm{ics}(\Delta_2)$. ∎

The following theorem tells us when and how we can transform from database $\Delta_1 = \langle \alpha, \mathcal{F}, \mathcal{B} \rangle$ to database $\Delta_2 = \langle \alpha, \mathcal{F}, \mathcal{C} \rangle$ without loss of information.

Theorem 4.2

If $\mathrm{ics}(\Delta_2) \leq \mathrm{ics}(\Delta_1)$, then the following transformation yields Δ_2 from Δ_1:

$$S(\gamma) = (\mathop{*}_{\beta \in \mathcal{B}(\gamma)} R_\beta)_\gamma$$

for each $\gamma \in \mathcal{C}$.

$$\mathrm{ics}(\Delta_1) = \{EDPL\}$$

$$\mathrm{ics}(\Delta_2) = \{DP, EPL\}$$

Figure 3. Information constraint sets. The notation $\{\{D,P\},\{E,P,L\}\}$ is simplified to $\{DP, EPL\}$.

4.2. Equivalence of Classes of Databases

We will say that two databases are equivalent if they are capable of storing the same information. The information constraint set characterizes the information that may be stored in a database, in the following sense:

> A relation $R(\alpha)$ can be derived from the database Δ if and only if α is contained in some member if $\mathrm{ics}(\Delta)$.

Thus two databases are capable of storing the same information if they have the same information constraint set. The example databases Δ_1 and Δ_2 do not store the same information, and as shown in figure 3, they do not have the same ics.

This observation motivates the following definition:

> Definition. Databases $\Delta_1 = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$ and $\Delta_2 = \langle \alpha, \mathcal{F}, \mathcal{B} \rangle$ are equivalent (written $\Delta_1 \cong \Delta_2$) if $\mathrm{ics}(\Delta_1) = \mathrm{ics}(\Delta_2)$.

Using the example developed in section 2, the following databases are all equivalent. (For the sake of notational compactness, the set $\{E,D,P\}$ is written EDP when it is a member of a database schema. Thus the schema $\{\{E,D\}, \{E,P,L\}\}$ is written $\{ED,EPL\}$.)

$$\Delta_n = \langle \alpha, \mathcal{F}, \mathcal{A}_n \rangle$$

with

$$\mathcal{A}_1 = \{EDPL\}$$

$$\mathcal{A}_2 = \{EDP,EDL,EPL,DPL\}$$

$$\mathcal{A}_3 = \{EDP,EPL,EPL\}$$

In fact, these are equivalent to all databases $\Delta = \langle \alpha, \mathcal{F}, B \rangle$ satisfying one of the following inequalities:

$$\{ED,EPL\} \leq \mathcal{B} \leq \{EDPL\}$$

or

$$\{DL,EPL\} \leq \mathcal{B} \leq \{EDPL\}$$

or

$$\{ED,DL,PL\} \leq \mathcal{B} \leq \{EDPL\}.$$

All of these databases have $\mathrm{ics}(\Delta) = \{EDPL\}$.

4.3. The Canonical Representation of an Equivalence Class

The information constraint set characterizes all of the database systems in an equivalence class. Thus it is an obvious candidate for the canonical representation.

> Definition. The canonical database in an equivalence class of databases containing the database $\Delta = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$ is the database
>
> $$\langle \alpha, \mathcal{F}, \mathrm{ics}(\Delta) \rangle.$$

The canonical database is well-defined because all of the databases in an equivalence class have the same information constraint set. Furthermore, it can be constructed from any member of the equivalence class, without any additional information about the equivalence class (see section 6). Thus $\langle \alpha, \mathcal{F}\{EDPL\}\rangle$ would be the canonical database for the equivalence class described at the end of the preceding section.

5. TRANSFORMATIONS

In this section, it is shown that the family of information constraint sets over a given set of attributes α and full family of functional dependencies $\mathcal{F}$, together with the partial order $\textcircled{\le}$, is a lattice.

This result allows us to define a transformation between database which are not comparable under $\textcircled{\le}$; because they are not comparable, information will be lost, but in fact, under the transformation, information loss is minimized. The loss is minimized because the transformation uses the greatest lower bound of the information constraint sets of the source and target databases.

The proof that the family of information constraint sets is a lattice is given in section 5.1. The transformation is described in section 5.2.

5.1. The Lattice

First, we define the greatest lower bound and prove its uniqueness.

<u>Definition.</u> For any collection $\mathcal{B}$ of sets,

$\max(\mathcal{B}) = \{\beta \in \mathcal{B} \mid \text{for all } \gamma \in \mathcal{B},\ \beta \subseteq \gamma \text{ implies } \beta = \gamma\}$

$\max(\mathcal{B})$ is the set of maximal subsets of $\mathcal{B}$.

<u>Definition</u>. Let $\mathcal{A}$, $\mathcal{B}$ be collections of subsets of α. Then define $\mathcal{A} \wedge \mathcal{B} = \max\{\alpha \cap \beta \mid \alpha \in \mathcal{A} \text{ and } \beta \in \mathcal{B}\}$.

$\mathcal{A} \wedge \mathcal{B}$ will be shown to be the greatest lower bound of $\mathcal{A}$ and $\mathcal{B}$ in the lattice of information constraint sets.

<u>Lemma</u> (GLB)

Let $\mathcal{I}_1 = \text{ics}(\Delta_1)$ and $\mathcal{I}_2 = \text{ics}(\Delta_2)$ where $\Delta_1 = \langle\alpha,\mathcal{F},\mathcal{B}\rangle$ and $\Delta_2 = \langle\alpha,\mathcal{F},\mathcal{C}\rangle$. Then $\mathcal{I}_1 \wedge \mathcal{I}_2$ is the information constraint set of $\langle\alpha,\mathcal{F},\mathcal{I}_1\wedge\mathcal{I}_2\rangle$. If $\Delta_3 = \langle\alpha,\mathcal{F},\mathcal{D}\rangle$ is any database with

$$\text{ics}(\Delta_3) \textcircled{\le} \mathcal{I}_1$$

and

$$\text{ics}(\Delta_3) \textcircled{\le} \mathcal{I}_2$$

then

$$\text{ics}(\Delta_3) \textcircled{\le} \mathcal{I}_1 \wedge \mathcal{I}_2 .$$

Proof:

$\text{ics}(\Delta_3) \textcircled{\le} \mathcal{I}_1$ and $\text{ics}(\Delta_3) \textcircled{\le} \mathcal{I}_2$ implies $\text{ics}(\Delta_3) \textcircled{\le} \mathcal{I}_1 \wedge \mathcal{I}_2$ from the definition of $\textcircled{\le}$ and $\wedge$.

We need only to show that $\mathcal{I}_1 \wedge \mathcal{I}_2$ is an ics.

Condition (1) follows immediately from the definition

Condition (2) is trivial:

$\gamma \in \mathcal{I}_1\wedge\mathcal{I}_2$ implies $R_\gamma \supseteq (\mathop{*}\limits_{\beta\in(\mathcal{I}_1\wedge\mathcal{I}_2)(\gamma)} R_\beta)_\gamma$. But by lemma 2.2,

$$R_\gamma \subseteq (\mathop{*}\limits_{\beta\in(\mathcal{I}_1\wedge\mathcal{I}_2)(\gamma)} R_\beta)_\gamma \quad .$$

Therefore $R_\gamma = (\mathop{*}\limits_{\beta\in(\mathcal{I}_1\wedge\mathcal{I}_2)(\gamma)} R_\beta)_\gamma$

To prove condition (2):

Suppose $R_\gamma = (\mathop{*}\limits_{\beta\in(\mathcal{I}_1\wedge\mathcal{I}_2)(\gamma)} R_\beta)_\gamma$.

$$\beta \in (\mathcal{I}_1 \wedge \mathcal{I}_2)(\gamma) \text{ implies } \beta = \alpha_1 \cap \alpha_2$$

for some $\alpha_1 \in \mathcal{I}_1$, $\alpha_2 \in \mathcal{I}_2$. Therefore

$$R_\gamma = (\mathop{*}\limits_{\alpha_1\in\mathcal{I}_1(\gamma)} (\mathop{*}\limits_{\alpha_2\in\mathcal{I}_2(\gamma)\cap\mathcal{I}_2(\alpha_1)} R_{\alpha_1\cap\alpha_2}))_\gamma$$

It follows from this and lemma 2.3 that

$$R_\gamma = (\mathop{*}\limits_{\alpha_1\in\mathcal{I}_1(\gamma)} (\mathop{*}\limits_{\alpha_2\in\mathcal{I}_2(\gamma)\cap\mathcal{I}_2(\alpha_1)} R_{\alpha_1}))_\gamma = (\mathop{*}\limits_{\alpha_1\in\mathcal{I}_1(\gamma)} R_{\alpha_1})_\gamma$$

But then $\gamma \subseteq \alpha_1$ for some $\alpha_1 \in \mathcal{I}_1$. Similarly, $\gamma \subseteq \alpha_2$ for some $\alpha_2 \in \mathcal{I}_2$.

Therefore $\gamma \subseteq \alpha_1 \cap \alpha_2 \subseteq \beta \in \mathcal{I}_1 \wedge \mathcal{I}_2$.

Given the set $\alpha = \{E,P,L,D\}$ of attributes, several different lattices can be defined, one for each full family of functional dependencies over α. For example, consider the case $\mathcal{F} = \emptyset$. The family of information constraint sets is then just the family of sets $\{\max(\mathcal{A}) | \mathcal{A}$ is a database schema$\}$. If $|\alpha| = n$, then this is the free distributive lattice over n elements ($F_D(n)$) together with the database schema $\{\{\alpha\}\}$.

5.2. The Transformation Algorithm

We will assume that we are given the lattice of equivalence classes of database systems for a set of attributes α and a full family of functional dependencies $\mathcal{F}$. The transformation algorithm will then be the composition of the following four transformations:

(1) The transformation from the source database to the canonical source database;

(2) The transformation from the canonical source database to the glb of the canonical source and the canonical target database;

(3) The transformation from the glb to the canonical target database; and

(4) The transformation from the canonical target database to the target database.

Thus two classes of transformations need to be defined:

(1) Transformations between a database and the canonical database for its equivalence class; and

(2) Transformations between comparable canonical databases.

Condition (2) of the definition of information constraint sets gives the transformation from a database $\Delta = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$ to the canonical database $\langle \alpha, \mathcal{F}, ics(\Delta) \rangle$ for its equivalence class:

$$\text{For each } \beta \in ics(\Delta),\ S(\beta) = \Big(\mathop{*}_{\gamma \in \mathcal{A}(\beta)} R(\gamma) \Big)_{\beta} .$$

Conversely, by Lemma 4.1, if $\Delta = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$, then $\mathcal{A} \leq \text{ics}(\Delta)$. Therefore, if $\beta \in \mathcal{A}$, then for some $\gamma \in \text{ics}(\Delta)$, we have $\beta \subseteq \gamma$. This gives the transformation from the canonical database for an equivalence to a database belonging to the equivalence class:

For each $\beta \in \mathcal{A}$, take $\gamma \in \text{ics}(\Delta)$ with $\beta \subseteq \gamma$;

then $S(\beta) = R(\gamma)_{\beta}$.

The second class of transformations, those between comparable canonical databases, is even simpler. Suppose we have $\Delta_1 = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$ and $\Delta_2 = \langle \alpha, \mathcal{F}, \mathcal{B} \rangle$ with $\text{ics}(\Delta_1) \leq \text{ics}(\Delta_2)$. Then to transform from $\langle \alpha, \mathcal{F}, \text{ics}(\Delta_1) \rangle$ to $\langle \alpha, \mathcal{F}, \text{ics}(\Delta_2) \rangle$.

For each $\beta \in \text{ics}(\Delta_2)$, set $S(\beta) = \bigcup_{\substack{\gamma \in \text{ics}(\Delta_1) \\ \beta \cap \gamma \neq \emptyset}} R(\gamma)^{\beta}$

where $R(\gamma)^{\beta}$ is defined as follows

$x \in R(\gamma)^{\beta}$ iff $x_{\gamma} \in R(\gamma)$

x_A undefined for $A \in \beta - \gamma$

Finally, to transform from $\langle \alpha, \mathcal{F}, \text{ics}(\Delta_2) \rangle$ to $\langle \alpha, \mathcal{F}, \text{ics}(\Delta_1) \rangle$:

For each $\beta \in \text{ics}(\Delta_1)$, choose $\gamma \in \text{ics}(\Delta_2)$

with $\beta \subseteq \gamma$.

Then $S(\beta) = R(\gamma)_{\beta}$.

The entire transformation algorithm from $\Delta_1 = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$ to $\Delta_2 = \langle \alpha, \mathcal{F}, \mathcal{B} \rangle$ can then be defined as follows:

T1. $[\Delta_1 \rightarrow \langle \alpha, \mathcal{F}, \text{ics}(\Delta_1) \rangle]$

For each $\gamma \in \text{ics}(\Delta_1)$ compute $S(\gamma) = [\mathop{*}_{\beta \in \mathcal{A}(\gamma)} R(\beta)]_{\gamma}$

T2. $[\langle\alpha,\mathcal{F},\mathrm{ics}(\Delta_1)\rangle \rightarrow \langle\alpha,\mathcal{F},\mathrm{ics}(\Delta_1) \wedge \mathrm{ics}(\Delta_2)\rangle]$

For each $\delta \in \mathrm{ics}(\Delta_1) \wedge \mathrm{ics}(\Delta_2)$, take $\gamma \in \mathrm{ics}(\Delta_1)$ such that $\delta \subseteq \gamma$ and compute $T(\delta) = S(\gamma)_\delta$.

T3. $[\langle\alpha,\mathcal{F},\mathrm{ics}(\Delta_1) \wedge \mathrm{ics}(\Delta_2)\rangle \rightarrow \langle\alpha,\mathcal{F},\mathrm{ics}(\Delta_2)\rangle]$

For each $\varphi \in \mathrm{ics}(\Delta_2)$, compute $Q(\varphi) = \bigcup_{\substack{\delta\in\mathrm{ics}(\Delta_1)\wedge\mathrm{ics}(\Delta_2)\\ \delta \cap \varphi \neq \emptyset}} T(\delta)^{\varphi}$.

T4. $[\langle\alpha,\mathcal{F},\mathrm{ics}(\Delta_2)\rangle \rightarrow \Delta_2]$

For each $\beta \in \mathcal{B}$, take $\varphi \in \mathrm{ics}(\Delta_2)$ with $\beta \subseteq \varphi$ and compute $R(\beta) = Q(\varphi)_\beta$.

6. DISCUSSION

The transformation algorithm assumes that the lattice of equivalence classes of database systems has been constructed. In this section, we discuss construction of the lattice.

In the case that the attribute set α has 3 or fewer members, the construction of the lattice is a trivial consequence of Theorem 3.1 (see section 6.1).

The construction is more difficult for all larger attribute sets. It would be simplified by a general decomposition theorem for relations, in the form:

> Let α be a set of attributes. Let $\mathcal{F}$ be a full family of functional dependencies over α. Then
>
> $$R(\alpha) = R(\alpha)_{\alpha(1)} * \ldots * R(\alpha)_{\alpha(n)}$$
>
> for all $R \in \mathcal{R}(\mathcal{F})$ if and only if $P(\alpha(1),\ldots,\alpha(n))$.

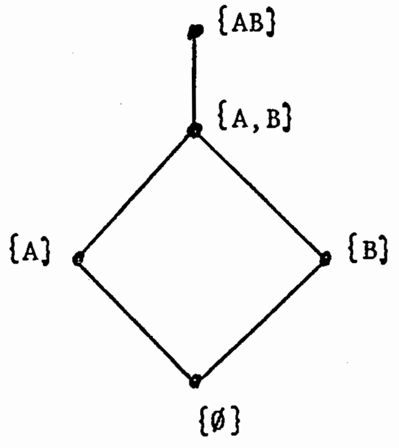
$P(\alpha(1),\ldots,\alpha(n))$ gives the necessary and sufficient conditions on $\alpha(1),\ldots,\alpha(n)$.

With such a theorem, we could say that $\mathcal{A} = \{\alpha(1),\ldots,\alpha(n)\}$ is the information constraint set characteristic of an equivalence class if and only if for every $\beta \subseteq \alpha$, if $P(\alpha(1) \cap \beta,\ldots,\alpha(n) \cap \beta)$ then $\beta \subseteq \alpha(i)$ for some i.

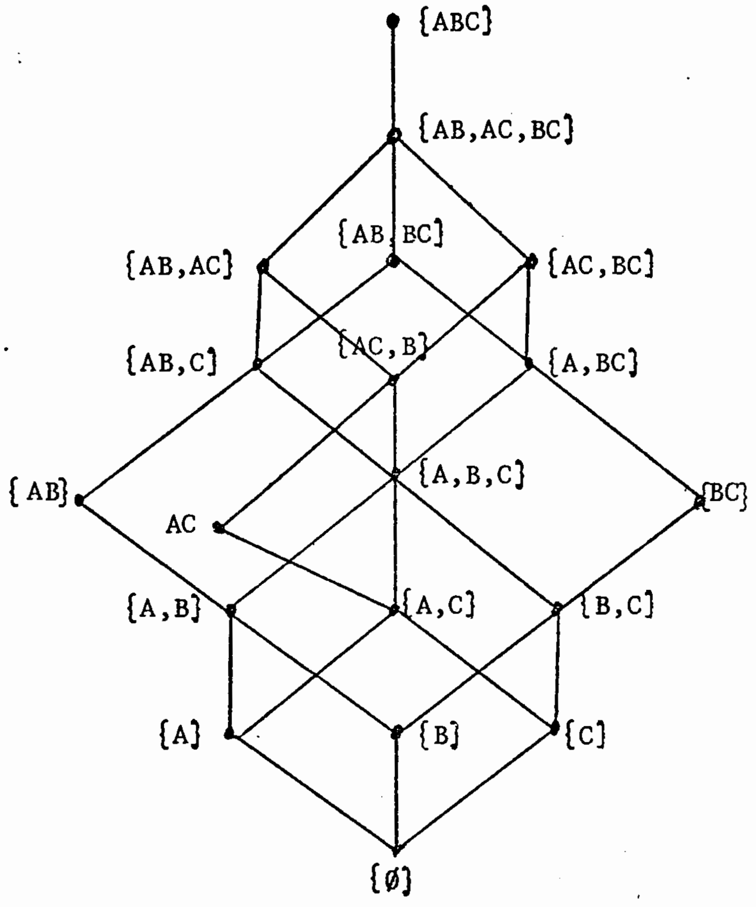
The crucial step of an algorithm to construct the lattice would then set $\mathcal{A}$ to $\max(\mathcal{A} \cup \{\beta\})$ for each $\mathcal{A} = \{\alpha(1),\ldots,\alpha(n)\}$ such that $P(\alpha(1) \cap \beta,\ldots,\alpha(n) \cap \beta)$. This would be repeated until there are no more schemas $\mathcal{A}$ with $P(\alpha(1) \cap \beta,\ldots,\alpha(n) \cap \beta)$.

6.1. The Lattices for α with 2 or 3 Members

The lattices for $\alpha = \{A,B\}$ and $\alpha = \{A,B,C\}$ and $\mathcal{F} = \emptyset^*$ are diagrammed in Figure 4. For $\mathcal{F} = \emptyset^*$, the lattice of equivalence classes of database systems is just the lattice of database systems. Thus there is only one database system in each equivalence class, and there is one node of the diagram for each database schema.



(a)



(b)

Figure 4

(a) The lattice for $\alpha = \{A,B\}$

(b) The lattice for $\alpha = \{A,B,C\}$

If $\alpha = \{A,B\}$, then the lattice is the same for all $\mathcal{F}$. This is because $R_A * R_B = R_A \times R_B$, and for $R = \{\langle 0,0\rangle,\langle 1,1\rangle\}$, $R_A \times R_B = \{\langle 0,0\rangle,\langle 0,1\rangle,\langle 1,0\rangle,\langle 1,1\rangle\}$. Note that $R \in \mathcal{R}(\mathcal{F})$ for all $\mathcal{F}$; so $\{A,B\} \not\trianglerighteq \{AB\}$ for any $\mathcal{F}$. This observation can be extended to any $\mathcal{A} = \{\alpha(1),\ldots,\alpha(n)\}$ with $\alpha(i) \cap \alpha(j) = \emptyset$ for all i,j (we will call such schemas <u>disjunctive</u>); for

$$R_{\alpha(1)} * \ldots * R_{\alpha(n)} = R_{\alpha(1)} \times \ldots \times R_{\alpha(n)}$$

and if we set $R = \{\langle 0,\dots,0\rangle,\langle 1,\dots,1\rangle\}$, then for some $x \in R_{\alpha(1)} \times \dots \times R_{\alpha(n)}$, $x_{\alpha(1)} = \langle 0,\dots,0\rangle$ and $x_{\alpha(2)} = \langle 1,\dots,1\rangle$. Therefore $x \notin R$.

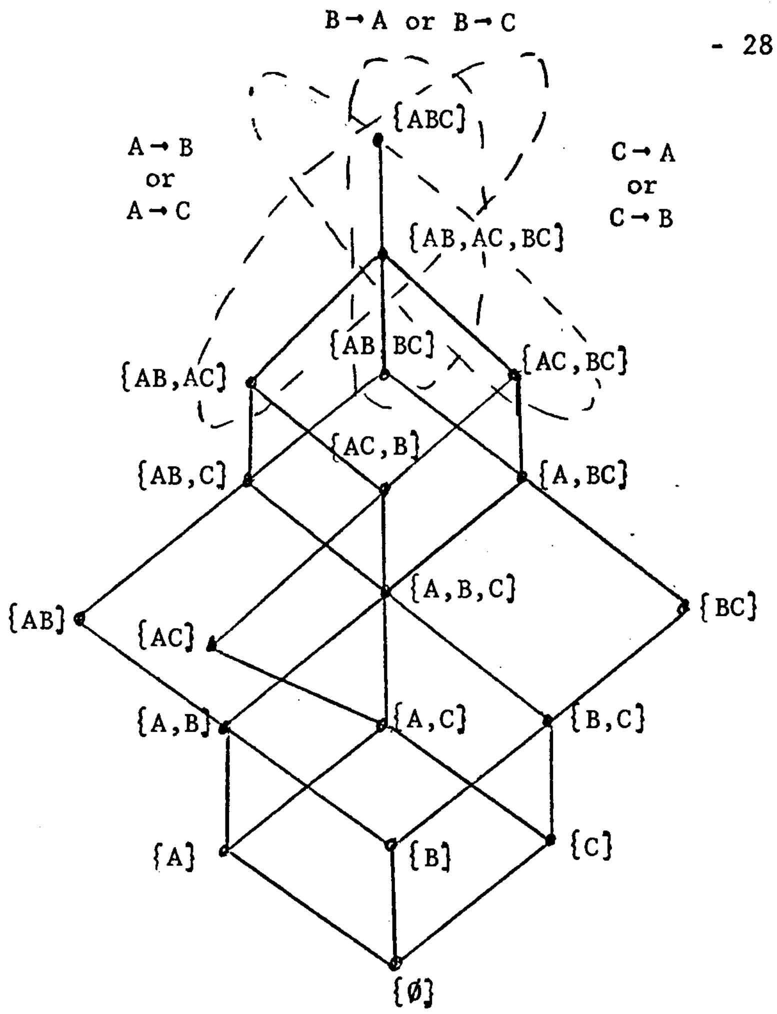
Figure 5 illustrates the lattices corresponding to various $\mathcal{F}$ over $\alpha = \{A,B,C\}$. Since all schemas in the lower half of the lattice are disjunctive, we ignore this part of the lattice after part (a). Theorem 3.1 together with Lemma 2.4 implies that

$$R = R_{AB} * R_{AC} = R_{AB} * R_{AC} * R_{BC} \quad \text{iff} \quad A \to B \text{ or } A \to C$$

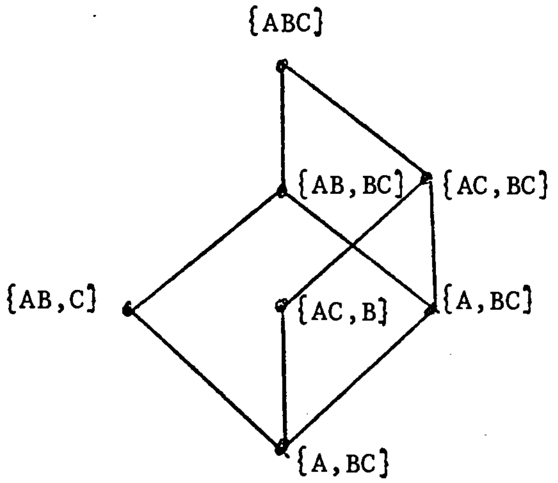
$$R = R_{AB} * R_{AC} = R_{AB} * R_{AC} * R_{BC} \quad \text{iff} \quad B \to A \text{ or } B \to C$$

$$R = R_{AC} * R_{BC} = R_{AB} * R_{AC} * R_{BC} \quad \text{iff} \quad C \to A \text{ or } C \to B .$$

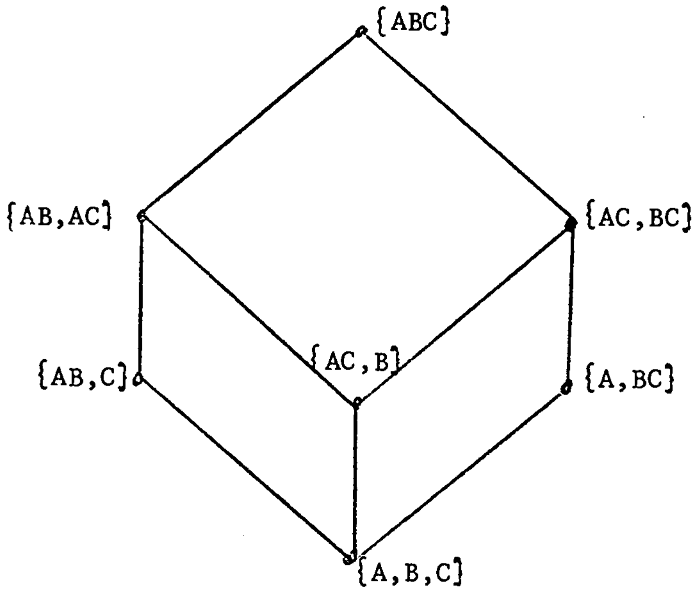
Therefore (as illustrated in Figure 5) the lattice depends on which of the above mentioned functional dependencies belong to $\mathcal{F}$.



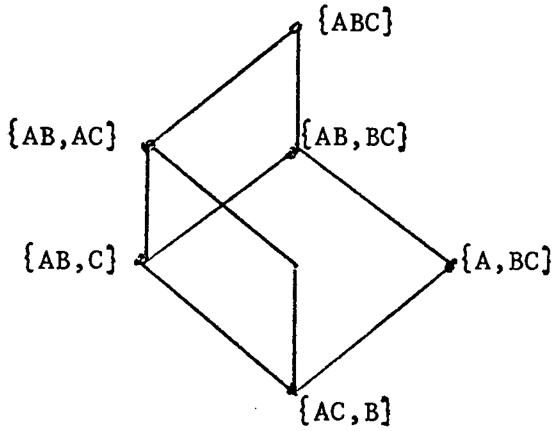
(a)



(b)



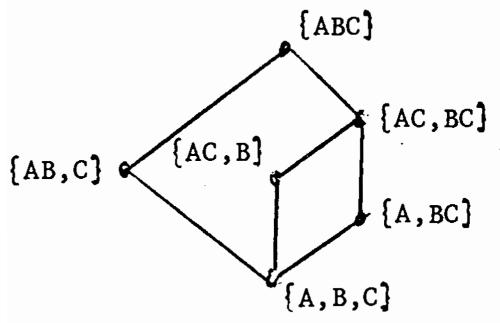
(c)



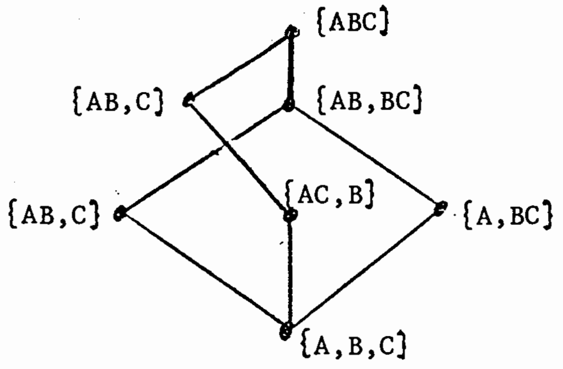
(d)

Figure 5

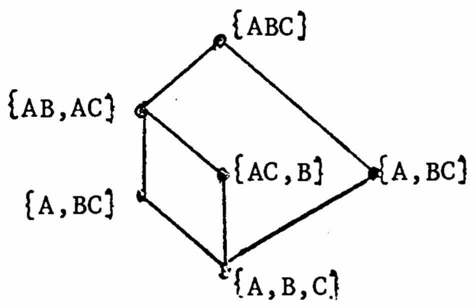
Figure 5 (continued)



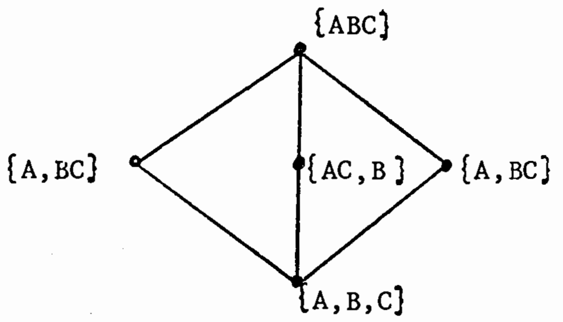
(e)



(f)



(g)



(h)

Figure 5

(a) The lattice for $\mathcal{F} = \emptyset^*$. The nodes surrounded by dotted lines collapse into one equivalence class if $\mathcal{F}$ contains any of the indicated functional dependencies.

(b) $\mathcal{F}$ contains $A \to B$ or $A \to C$ but not $B \to A$, $B \to C$, $C \to A$ or $C \to B$.

(c) $\mathcal{F}$ contains $B \to A$ or $B \to C$ but not $A \to B$, $A \to C$, $C \to A$ or $C \to B$.

(d) $\mathcal{F}$ contains $C \to A$ or $C \to B$ but not $A \to B$, $A \to C$, $B \to A$ or $B \to C$.

(e) $\mathcal{F}$ contains $A \to B$ or $A \to C$; and $B \to A$ or $B \to C$; but not $C \to A$ or $C \to B$.

(f) $\mathcal{F}$ contains $A \to B$ or $A \to C$; and $C \to A$ or $C \to B$; but not $B \to A$ or $B \to C$.

(g) $\mathcal{F}$ contains $B \to A$ or $B \to C$; and $C \to A$ or $C \to B$; but not $A \to B$ or $A \to C$.

(h) $\mathcal{F}$ contains $A \to B$ or $A \to C$; and $B \to A$ or $B \to C$; and $C \to A$ or $C \to B$.

6.2. The Lattice for Larger α.

The lattice is much more difficult to construct when the set α of attributes has more than three members. If $|\alpha| = 4$ and $\mathcal{F} = \emptyset^*$, then the lattice has 167 nodes. Rather than displaying a diagram of the entire lattice, its nodes are listed in table 6, and some sublattice diagrams are given in figure 7. Also, the nodes for $\mathcal{F}$ that belong to the same equivalence class for $\mathcal{G} = \{E \rightarrow D,\ L \rightarrow D,\ EP \rightarrow L,\ DP \rightarrow L\}^*$ are indicated in both table 6 and figure 7. by assigning them the same node number.

In computing ics(Δ) for $\Delta = \langle \alpha, \mathcal{G}, \mathcal{a} \rangle$ and $\mathcal{G} = \{E \rightarrow D,\ L \rightarrow D,\ DP \rightarrow L,\ EP \rightarrow L\}^*$, we must show:

(1) for each $\beta \in \text{ics}(\Delta)$, $\beta = (\underset{\alpha \in \mathcal{a}(\beta)}{*} R_\alpha)_\beta$

and

(2) if $\beta \notin \text{ics}(\Delta)$ and $\beta \not\subseteq \gamma$ for $\gamma \in \text{ics}(\Delta)$ then $R_\beta \neq (\underset{\alpha \in \mathcal{a}(\beta)}{*} R_\alpha)_\beta$ for some $R \in \mathcal{R}(\mathcal{F})$.

The third co lumn of table 6 gives the derivations required to show (1). The fourth column indicates how (2) may be shown:

<u>Case 1</u>: If there are only two relations in $\mathcal{a}$, use theorem 3.1.

<u>Case 2</u>: If $\mathcal{a}$ is disjunctive, then ics(Δ) = $\mathcal{a}$.

Case 3: If $\mathcal{a}$ is not disjunctive and there are more than two relations in $\mathcal{a}$, then (in the absence of a generalization of theorem 3.1) we must use a construction to show that (2) holds. One such construction is given here; the remainder are numerous but not difficult to construct. As the size of a increases, however, this construction becomes more difficult. An algorithm is given in "Decomposition and Recoverability of Relations" [2] to construct such a counter-example.

Let $\mathcal{A}$ = {EDP,EL,DL}. Then for the relation:

E	D	P	L
0	0	0	1
0	0	1	0
1	0	2	0

the projections are:

E	D	P
0	0	0
0	0	1
1	0	2

E	L
0	1
0	0
1	0

D	L
0	1
0	0

and the join is:

(E	D	P	*	E	L) *	D	L	=	E	D	P	L
0	0	0		0	1	0	1		0	0	0	1
0	0	0		0	0	0	0		0	0	0	0
0	0	1		0	1	0	1		0	0	1	1
0	0	1		0	0	0	0		0	0	1	0
1	0	2		1	0	0	0		1	0	2	0

The original relation was not recovered; in fact, only R_{EDP} and R_{EDL} were recovered, so that {EDP,EDL} is the set of maximal recoverable relations.

Node Number in $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$	Node ics in $\Delta = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$	Derivation of ics in $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$		Proof that ics for $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$ is maximal
1	EDPL			
1	EDP,EDL,EPL,DPL	$R_{EDL}*R_{EPL}$	(EL→D)	
1	EDP,EDL,EPL	$R_{EDL}*R_{EPL}$	(EL→D)	
1	EDP,EDL,DPL	$R_{EDP}*R_{DPL}$	(DP→L)	
1	EDP,EPL,DPL	$R_{EDP}*R_{DPL}$	(DP→L)	
1	EDL,EPL,DPL	$R_{EDL}*R_{E}$	(EL→D)	
1	EDP,EDL,PL	$R_{EDP}*(R_{DL}*R_{PL})$	(L→D,DP→L)	
1	EDP,EPL,DL	$R_{EDP}*R_{EPL}$	(EP→L)	
1	EDL,EPL,DP	$R_{EDL}*R_{EPL}$	(EL→D)	
1	EDP,DPL,EL	$R_{EDP}*R_{DPL}$	(DP→L)	
1	EDL,DPL,EP	$(R_{ED}*R_{EP})*R_{DPL}$	(E→D,DP→L)	
1	EPL,DPL,ED	$R_{EPL}*R_{ED}$	(E→D)	
2	EDP,EDL			Case 1
1	EDP,EPL	$R_{EDP}*R_{EPL}$	(EP→L)	
1	EDL,EPL	$R_{EDL}*R_{EPL}$	(EL→D)	
1	EDP,EPL	$R_{EDP}*R_{DPL}$	(DP→L)	
3	EDL,DPL			Case 1
1	EPL,DPL	$R_{EPL}*R_{DPL}$	(PL→D)	
1	EDP,EL,DL,PL	$R_{EDP}*R_{DL}*R_{PL}$	(L→D,DP→L)	
1	EDL,EP,DP,PL	$(R_{DL}*R_{PL})*(R_{ED}*R_{EP})$	(L→D,E→D,DP→L)	
1	EPL,ED,DP,DL	$R_{EPL}*R_{ED}$	(E→D)	
1	DPL,ED,EP,EL	$R_{ED}*E_{EP}*R_{DPL}$	(E→D,DP→L)	
2	EDP,EL,DL	$R_{EL}*R_{DL}$	(L→D)	Case 3
1	EDP,EL,PL	$(R_{ED}*R_{EL})_{DL}*R_{PL})*R_{EDP}$	(E→D,L→D,DP→L)	
1	EDP,DL,PL	$R_{EDP}*R_{DL}*R_{PL}$	(L→D,DP→L)	

Node number in $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$	Node ics in $\Delta = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$	Derivation of ics in $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$		Proof that the ics for $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$ is maximal
2	EDL,EP,DP	$R_{ED}*R_{EP}$	(E→D)	Case 3
1	EDL,EP,PL	$(R_{ED}*R_{EP})*(R_{DL}*R_{PL})$	(E→D,L→D,DP→L)	
3	EDL,DP,PL	$R_{DL}*R_{PL}$	(L→D)	Case 3
1	EPL,ED,DP	$R_{EPL}*R_{ED}$	(E→D)	
1	EPL,ED,DL	$R_{EPL}*R_{ED}$	(E→D)	
1	EPL,DP,DL	$R_{EPL}*R_{DL}$	(L→D)	
1	DPL,ED,EP	$R_{DPL}*(R_{ED}*R_{EP})$	(E→D,DP→L)	
3	DPL,ED,EL	$R_{ED}*R_{EL}$	(E→D)	Case 3
1	DPL,EP,EL	$((R_{DL}*R_{EL})_{ED}*R_{EP})*R_{DPL}$	(L→D,E→P,DP→L)	
2	EDP,EL	$R_{ED}*R_{EL}$	(E→D)	Case 1
4	EDP,DL			Case 1
5	EDP,PL			Case 1
2	EDL,EP	$R_{ED}*R_{EP}$	(E→D)	Case 1
6	EDL,DP			Case 1
3	EDL,PL	$R_{DL}*R_{PL}$	(L→D)	Case 1
1	EPL,ED	$R_{EPL}*R_{ED}$	(E→D)	
7	EPL,DP			Case 1
1	EPL,DL	$R_{EPL}*R_{DL}$	(L→D)	
8	DPL,ED			Case 1
9	DPL,EP			Case 1
3	DPL,EL	$R_{DL}*R_{EL}$	(L→D)	Case 1
10	EDP,L			Case 2
11	EDL,P			Case 2
12	EPL,D			Case 2
13	DPL,E			Case 2

Node no. in $\Delta = \langle \alpha,\mathcal{S},\mathcal{A}\rangle$	Node ics in $\Delta = \langle \alpha,\mathcal{F},\mathcal{A}\rangle$	Derivation of ics in $\Delta = \langle \alpha,\mathcal{S},\mathcal{A}\rangle$		Proof that the ics for $\Delta = \langle \alpha,\mathcal{S},\mathcal{A}\rangle$ is maximal
14	EDP			Case 2
15	EDL			Case 2
16	EPL			Case 2
17	DPL			Case 2
1	ED,EP,EL,DP,DL,PL	$(R_{ED}*R_{EP})*(R_{DL}*R_{PL})$	(E→D,L→D,DP→L)	
2	ED,EP,EL,DP,DL	$R_{ED}*R_{EP},R_{ED}*R_{EL}$	(E→D)	Case 3
1	ED,EP,EL,DP,PL	$(R_{ED}*R_{EP})((R_{ED}*R_{EL})_{DL}*R_{PL})$	(E→D,L→D,DP→L)	
1	ED,EP,EL,DL,PL	$(R_{ED}*R_{EP})*(R_{DL}*R_{PL})$	(E→D,L→D,DP→L)	
1	ED,EP,DP,DL,PL	$(R_{ED}*R_{EP})*(R_{DL}*R_{PL})$	(E→D,L→D,DP→L)	
3	ED,EL,DP,DL,PL	$R_{ED}*R_{EL},R_{DL}*R_{PL}$	(E→D,L→D)	Case 3
1	EP,EL,DP,DL,PL	$R_{EP}*(R_{EL}*R_{DL})_{ED}*(R_{DL}*R_{PL})$	(L→D,E→D,DP→L)	
2	ED,EP,EL,DP	$R_{ED}*R_{EP},R_{ED}*R_{EL}$	(E→D)	Case 3
2	ED,EP,EL,DL	$R_{ED}*R_{EP},R_{ED}*R_{EL}$	(E→D)	Case 3
1	ED,EP,EL,PL	$(R_{ED}*R_{EP})*((R_{ED}*R_{EL})_{DL}*R_{PL})$	(E→D,L→D,DP→L)	
4	ED,EP,DP,DL	$R_{ED}*R_{EP}$	(E→D)	Case 3
5	ED,EP,DP,PL	$R_{ED}*R_{EP}$	(E→D)	Case 3
1	ED,EP,DL,PL	$(R_{ED}*R_{EP})*(R_{DL}*R_{PL})$	(E→D,L→D,DP→L)	
3	ED,EL,DP,DL	$R_{ED}*R_{EL},(R_{ED}*R_{EL})_{DL}*R_{PL}$	(E→D,L→D)	Case 3
3	ED,EL,DP,PL	$R_{ED}*R_{EL},(R_{ED}*R_{EL})_{DL}*R_{PL}$	(E→D,L→D)	Case 3
1	ED,EL,DL,PL	$R_{ED}*(R_{DL}*R_{PL})_{PD}*(R_{DL}*R_{PL})$	(L→D,E→D,PD→L)	
8	ED,DP,DL,PL	$R_{DL}*R_{PL}$	(L→D)	Case 3
2	EP,EL,DP,DL	$R_{EL}*R_{DL},R_{EP}*(R_{EL}*R_{DL})_{ED}$	(L→D,E→D)	Case 3
18	EP,EL,DP,PL			Case 3
1	EP,EL,DL,PL	$R_{EP}*(R_{EL}*R_{DL})_{ED}*(R_{DL}*R_{PL})$	(L→D,E→D,DP→L)	
9	EP,DP,DL,PL	$R_{DL}*R_{PL}$	(L→D)	Case 3
3	EL,DP,DL,PL	$R_{EL}*R_{DL},R_{DL}*R_{PL}$	(L→D)	Case 3

Node number for $\Delta = \langle \alpha, \mathscr{S}, \mathscr{A} \rangle$	Node ics for $\Delta = \langle \alpha, \mathscr{F}, \mathscr{A} \rangle$	Derivation for ics for $\Delta = \langle \alpha, \mathscr{S}, \mathscr{A} \rangle$		Proof that ics for $\Delta = \langle \alpha, \mathscr{S}, \mathscr{A} \rangle$ is maximal
10	ED,EP,DP,L	$R_{ED}{*}R_{EP}$	(E→D)	Case 3
11	ED,EL,DL,P	$R_{ED}{*}R_{EL}$	(E→D)	Case 3
19	EP,EL,PL,D			Case 3
13	DP,DL,PL,E	$R_{DL}{*}P_{PL}$	(L→D)	Case 3
2	ED,EP,EL	$R_{ED}{*}R_{EP}, R_{ED}{*}R_{EL}$	(E→D)	Case 3
14	ED,EP,DP	$R_{ED}{*}R_{EP}$	(E→D)	Case 3
4	ED,EP,DL	$R_{ED}{*}R_{EP}$	(E→D)	Case 3
5	ED,EP,PL	$R_{ED}{*}R_{EP}$	(E→D)	Case 3
6	ED,EL,DP	$R_{ED}{*}R_{EL}$	(E→D)	Case 3
15	ED,EL,EL	$R_{ED}{*}R_{EL}$	(E→D)	Case 3
3	ED,EL,PL	$R_{ED}{*}R_{EL}, (R_{ED}{*}R_{EL})_{DL}{*}R_{PL}$	(E→D,L→D)	Case 3
20	ED,DP,DL			Case 3
21	ED,DP,PL			Case 3
8	ED,DL,PL	$R_{DL}{*}R_{PL}$	(L→D)	Case 3
22	EP,EL,DP			Case 3
2	EP,EL,DL	$R_{ED}{*}(R_{EL}{*}R_{DL})_{ED}, R_{EL}{*}R_{DL}$	(L→D,E→D)	Case 3
23	EP,EL,PL			Case 3
24	EP.DP,EL			Case 3
25	EP,DP,PL			Case 3
9	EP,DL,PL	$R_{DL}{*}R_{PL}$	(L→D)	Case 3
6	EL,DP,DL	$R_{EL}{*}R_{DL}$	(L→D)	Case 3
26	EL,DP,PL			Case 3
3	EL,DL,PL	$R_{EL}{*}R_{DL}, R_{DL}{*}R_{PL}$	(L→D)	Case 3
17	DP,DL,PL	$R_{DL}{*}P_{PL}$	(L→D)	Case 3

Node number for $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$	Node ics for $\Delta = \langle \alpha, \mathcal{T}, \mathcal{A} \rangle$	Derivation of ics for $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$	Proof that ics for $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$
10	ED,EP,L	$R_{ED}{*}R_{EP}$ (E→D)	Case 3
11	ED,EL,P	$E_{ED}{*}R_{EL}$ (E→D)	Case 3
27	ED,DP,L		Case 3
28	ED,DL,P		Case 3
29	EP,EL,D		Case 3
30	EP,DP,L		Case 3
31	EP,PL,D		Case 3
11	EL,DL,P	$R_{EL}{*}R_{DL}$ (L→D)	Case 3
32	EL,PL,D		Case 3
33	DP,DL,E		Case 3
34	DP,PL,E		Case 3
13	DL,PL,E	$R_{DL}{*}R_{PL}$ (L→D)	Case 3
14	ED,EP	$R_{ED}{*}R_{EP}$ (E→D)	Case 1
15	ED,EL	$R_{ED}{*}R_{EL}$ (E→D)	Case 1
35	ED,DP		Case 1
36	ED,DL		Case 1
37	ED,PL		Case 2
38	EP,EL		Case 1
39	EP,DP		Case 1
40	EP,DL		Case 2
41	EP,PL		Case 1
42	EL,DP		Case 2
15	EL,DL	$R_{EL}{*}R_{DL}$ (L→D)	Case 1
43	EL,PL		Case 1
44	DP,DL		Case 1
45	DP,PL		Case 1
17	DL,PL	$R_{DL}{*}R_{PL}$ (L→D)	Case 1

Node number for $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$	Node ics for $\Delta = \langle \alpha, \mathcal{F}, \mathcal{A} \rangle$	Derivation of ics for $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$	Proof that ics for $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$ is maximal
46	ED,P,L		Case 1
47	EP,D,L		Case 1
48	EL,D,P		Case 1
49	DP,E,L		Case 1
50	DL,E,P		Case 1
51	PL,E,D		Case 1
52	ED,P		Case 1
53	ED,L		Case 1
54	EP,D		Case 1
55	EP,L		Case 1
56	EL,D		Case 1
57	EL,P		Case 1
58	DP,E		Case 1
59	DP,L		Case 1
60	DL,E		Case 1
61	DL,P		Case 1
62	PL,E		Case 1
63	PL,D		Case 1
64	ED		Case 1
65	EP		Case 1
66	EL		Case 1
67	DP		Case 1
68	DL		Case 1
69	PL		Case 1
70	E,D,P,L		Case 1
71	E,D.P		Case 1
72	E,D,L		Case 1
73	E,P.L		Case 1
74	D,P.L		Case 1
75	E,D		Case 1
76	E,P		Case 1
77	E,L		Case 1
78	D,P		Case 1
79	D,L		Case 1
80	P.L		Case 1
81	E		Case 1
82	D		Case 1
83	P		Case 1
84	L		Case 1

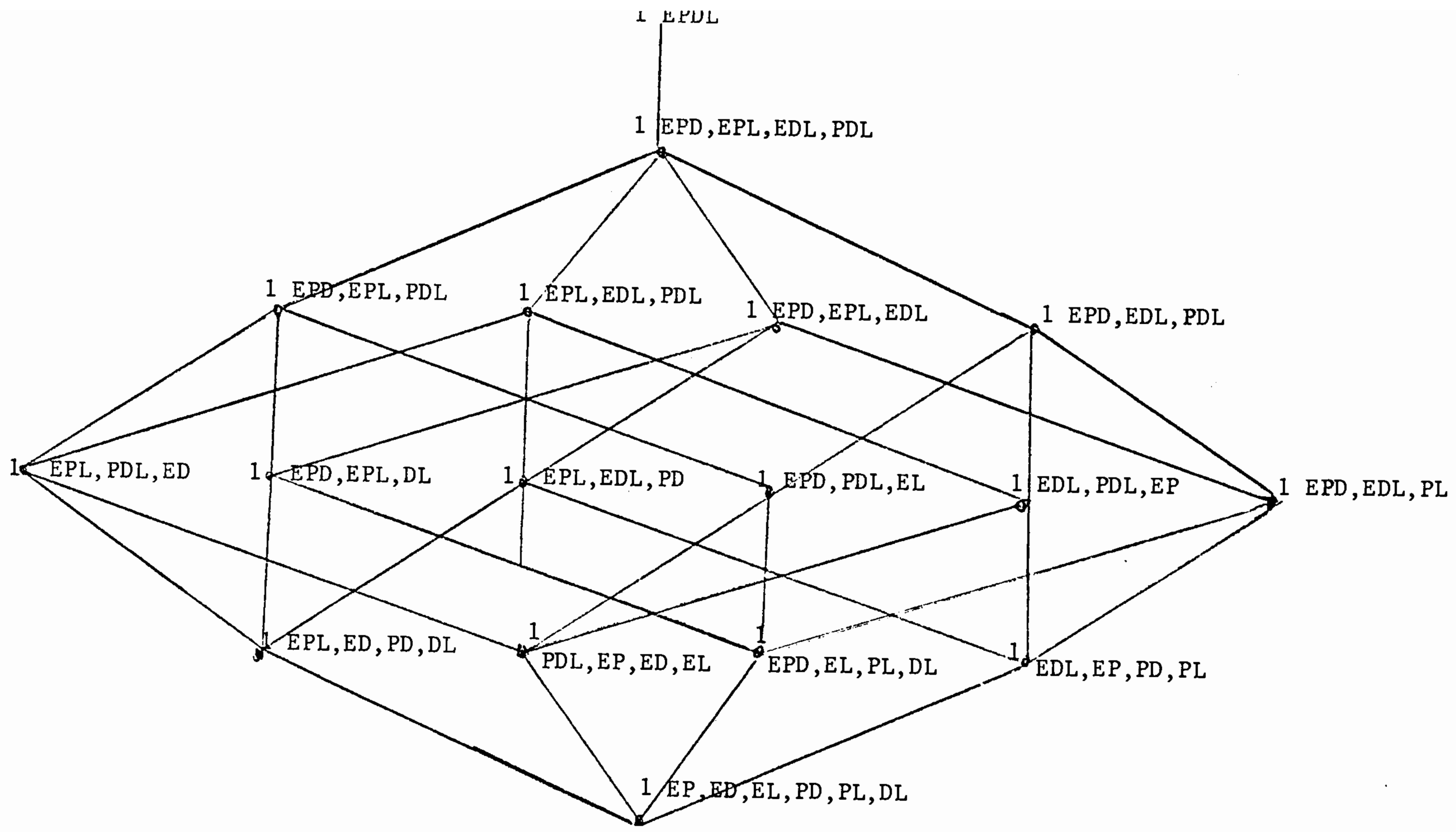


Figure 7a. A sublattice of the lattice of database equivalence classes for $\Delta = \langle \alpha, \emptyset^*, \mathcal{A} \rangle$, $\alpha = \{E,D,P.L\}$. This is the sublattice with $\{EP,ED,EL,PD,PL,DL\}$ $\textcircled{\leq}$ ics(Δ) $\overline{\textcircled{\leq}}$ $\{EDPL\}$. The numbers labelling the nodes give the database equivalence classes in $\Delta = \langle \alpha, \mathcal{B}, \mathcal{A} \rangle$ with $\alpha = \{E,D,P,L\}$ and $\mathcal{B} = \{E \rightarrow D,\ L \rightarrow D,\ DP \rightarrow L\}$.* (All database schemas $\mathcal{A}$ in this lattice give databases equivalent to $\langle \alpha, \mathcal{B}, \{EDPL\} \rangle$.)

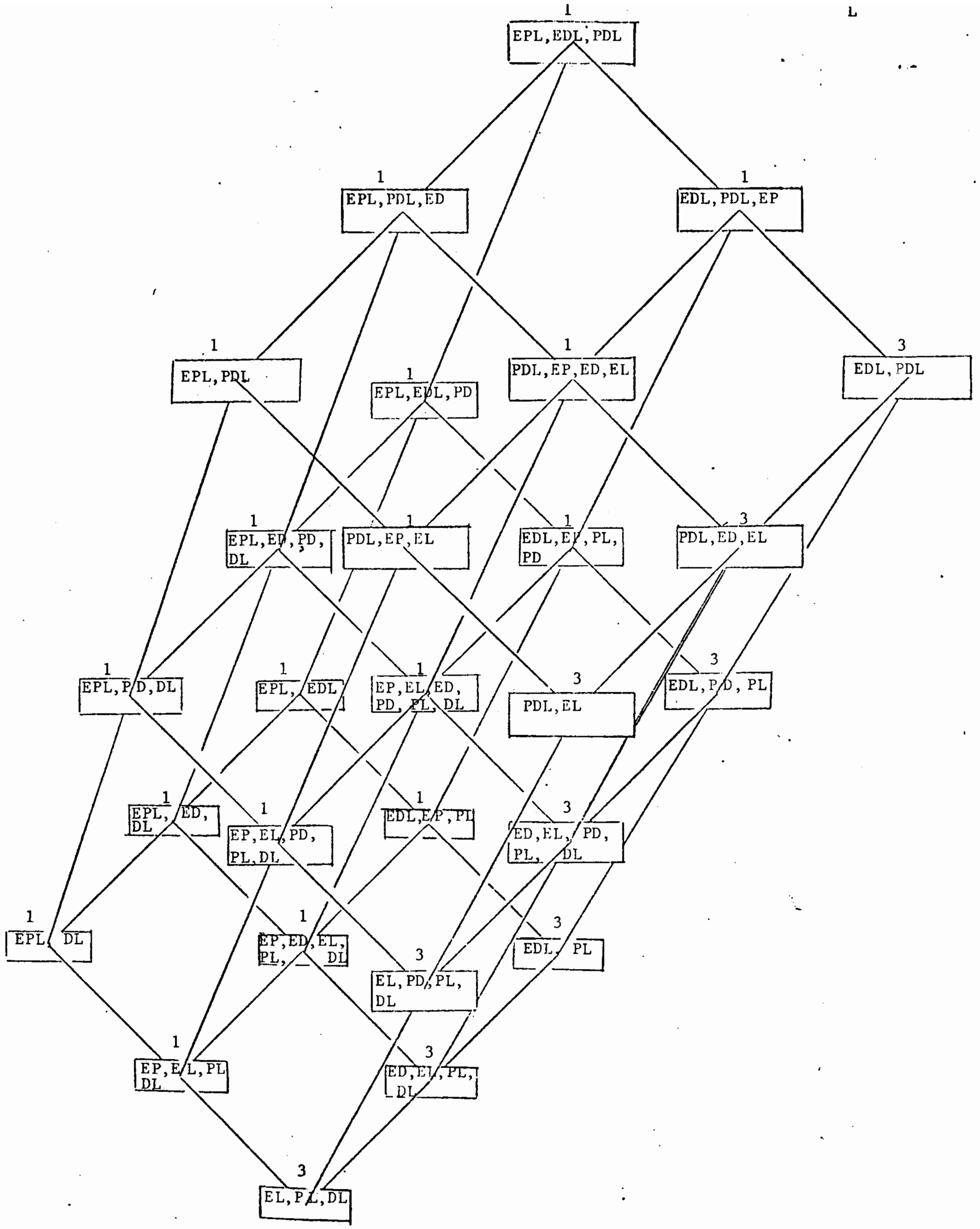


Figure 7b. A sublattice of the lattice of equivalence classes for $\Delta = \langle \alpha, \emptyset^*, \mathcal{A} \rangle$, $\alpha = \{E,D,P.L\}$. This is the sublattice with $\{EL,PL,DL\} \textcircled{\le} ics(\Delta) \textcircled{\le} \{EPL,EDL,PDL\}$. There are 3 isomorphic sub-lattices: $\{ED,EP,EL\} \textcircled{\le} ics(\Delta) \textcircled{\le} \{EDP,EPL,EDL\}$

$\{EP,DP,PL\} \textcircled{\le} ics(\Delta) \textcircled{\le} \{EDP,EPL,PDL\}$

$\{ED,DP,DL\} \textcircled{\le} ics(\Delta) \textcircled{\le} \{EDP,EDL,PDL\}$

(See figures 7c, 7d, and 7e). The numbers labelling the nodes indicate the database equivalence classes in $\Delta = \langle \alpha, \mathcal{S}, \mathcal{A} \rangle$, with $\alpha = \{E,D,P,L\}$ and $\mathcal{S} = \{E \to P, L \to D, DP \to L\}^*$

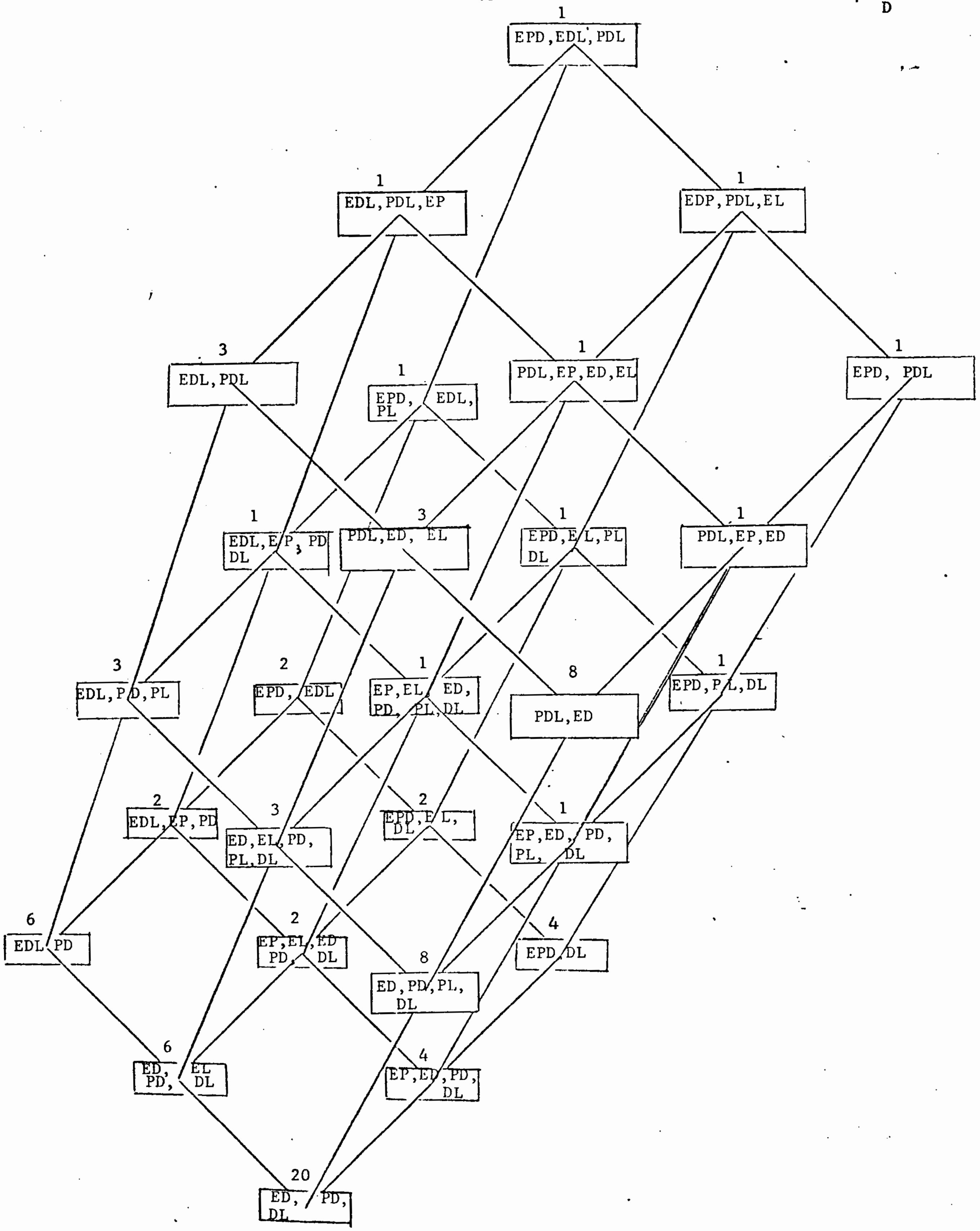


Figure 7c. The sublattice {ED,PD,DL} ⊆ ics(Δ) ⊆ {EPD,EDL,PDL} of the lattice of database equivalence classes for Δ = ⟨α,∅*,𝒜⟩, α = {E,D,P,L}.

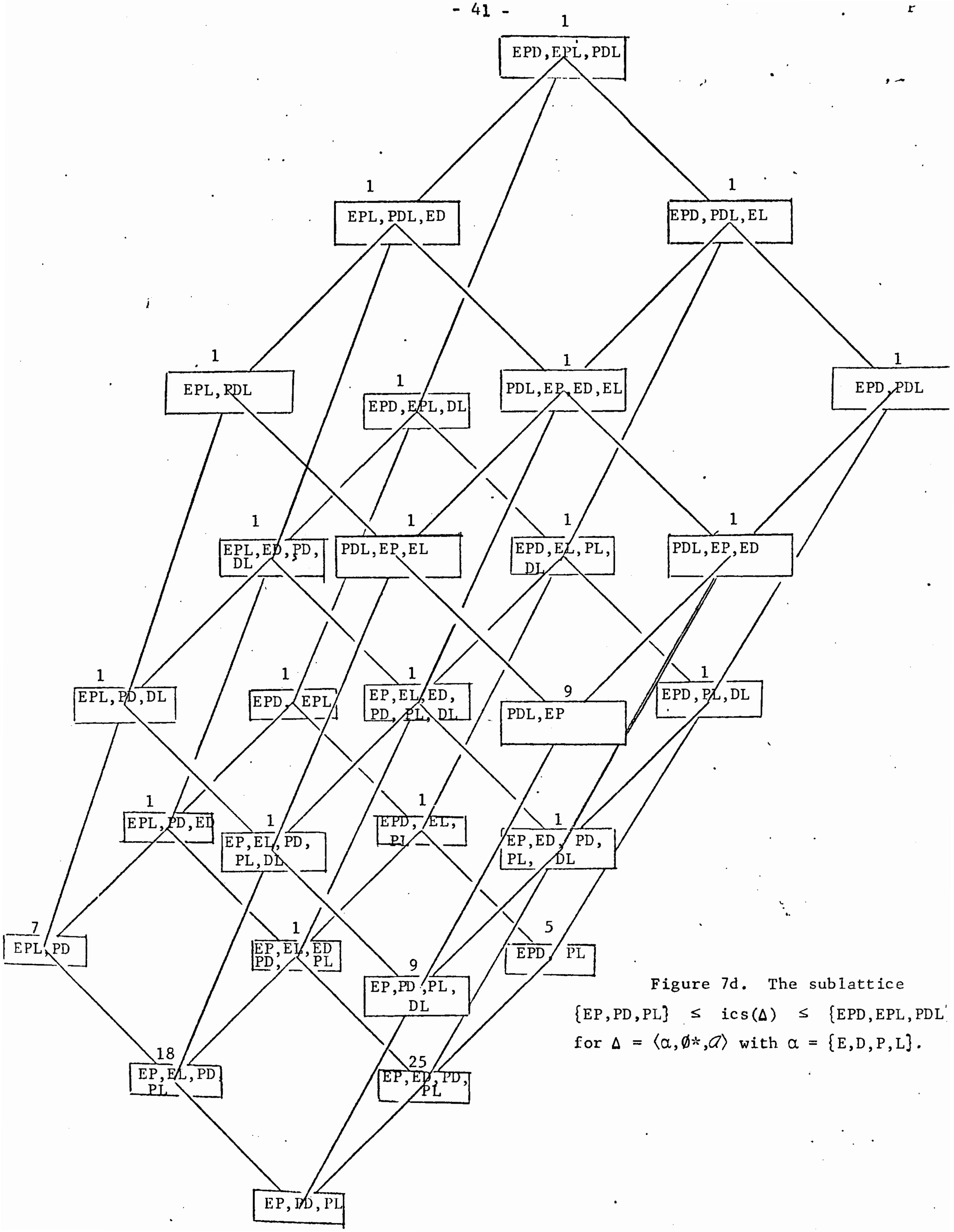


Figure 7d. The sublattice $\{EP,PD,PL\} \leq ics(\Delta) \leq \{EPD,EPL,PDL\}$ for $\Delta = \langle\alpha,\emptyset^*,\mathcal{A}\rangle$ with $\alpha = \{E,D,P,L\}$.

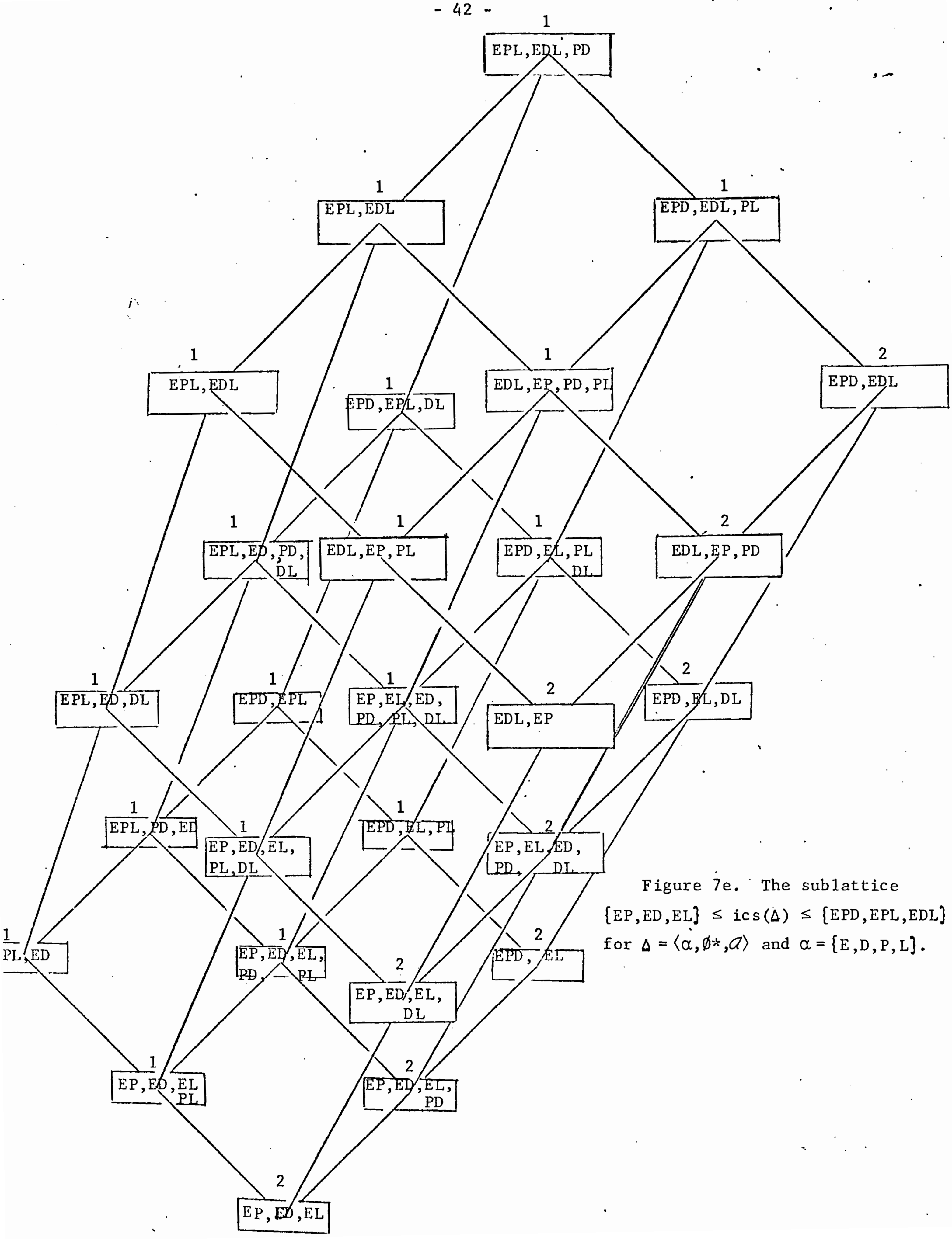


Figure 7e. The sublattice $\{EP,ED,EL\} \leq ics(\Delta) \leq \{EPD,EPL,EDL\}$ for $\Delta = \langle \alpha, \emptyset^*, \mathcal{A} \rangle$ and $\alpha = \{E,D,P,L\}$.

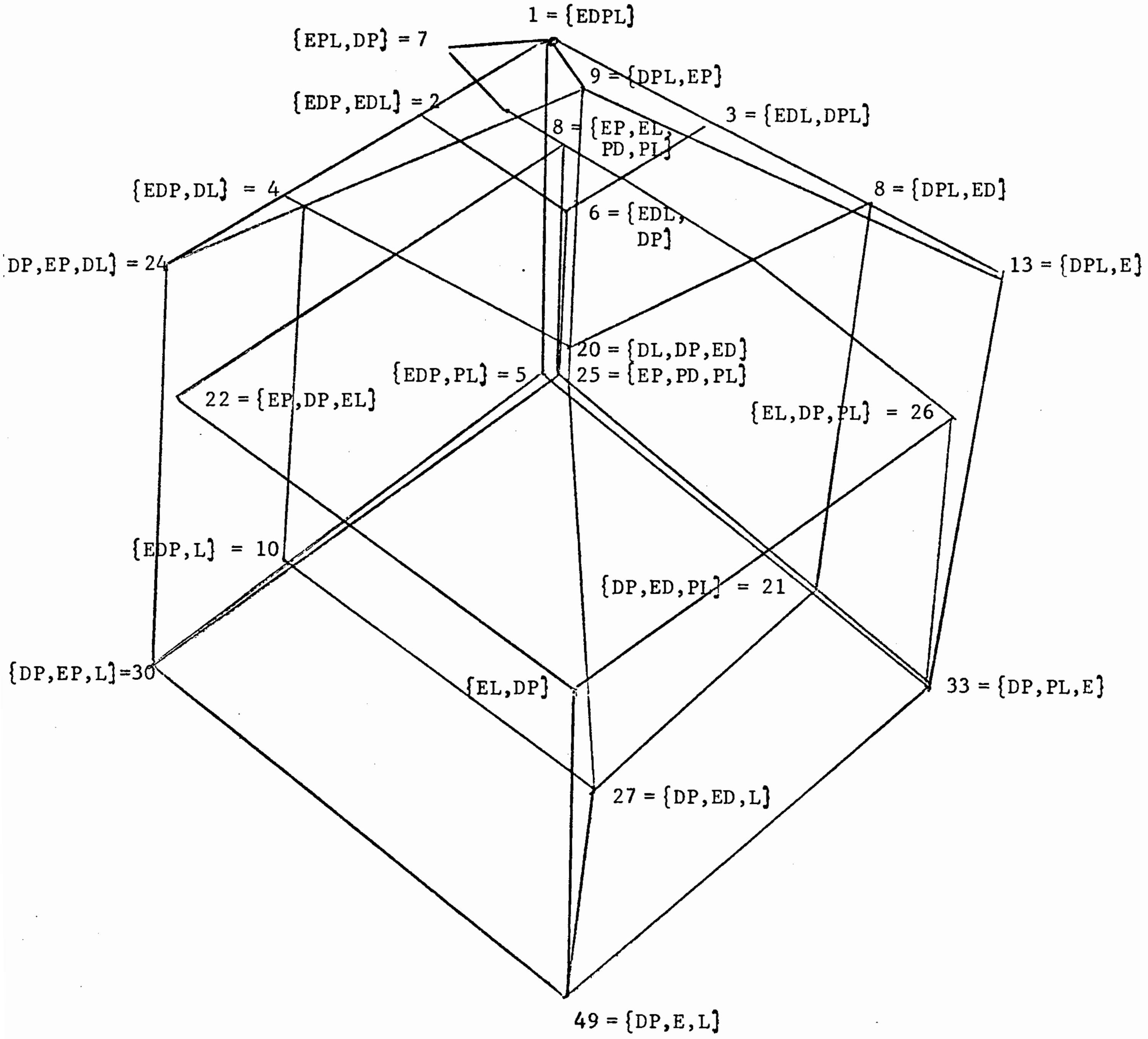


Figure 8. A sublattice of the lattice for $\Delta = \langle \alpha, \mathcal{B}, \mathcal{A} \rangle$ with $\alpha = \{E,D,P,L\}$ $\mathcal{B} = \{E \rightarrow D,\ L \rightarrow D,\ DP \rightarrow L\}^*$. The node numbers are taken from table 6.

REFERENCES

1. Armstrong, W.W., "Dependency Structures of Data Base Relationships", Information Processing, 74, North-Holland Publishing Co., Amsterdam, 1974, pp. 580-583.

2. Codd, E.F., "A Relational Model of Data for Large Shared Data Banks", Comm. ACM, 13, 6 (June 1970), 377-387.

3. Delobel, C., "Aspects Theoreques sur la Structure de l'Information dans Une Base de Donees", Revue Informatique et Recherche Operationnelle, B3 (1971), pp. 1017-1021.

4. Griffeth, N., Database Structures and Transformations, Ph.D. dissertation, University of Chicago, 1978.

5. Griffeth, N., "Decomposition and Recoverability of Relations", Dis- cussion Paper No. 354, Northwestern University, Center for Mathematical Studies in Economics and Management Science.

6. Heath, I.J., "Unacceptable File Operations in a Relational Data Base", Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access and Control, San Diego, CA, pp. 19-68.

7. Rissanen, J., "Independent Components of Relations", ACM Trans. Database Systems, Dec. 1977, v. 2, no. 4, pp. 317-325.