On Restructuring Nested Relations in Partitioned Normal Form

Guy Hulin

Philips Research Laboratory
Avenue Albert Einstein, 4
B-1348 Louvain-la-Neuve, Belgium
ghulin@prlb.philips.be

Abstract

Relations in partitioned normal form are an important subclass of nested relations. This paper is concerned with the problem of restructuring relations in partitioned normal form to new and potentially very different schemes. The main problem with restructuring is to minimize the amount of information lost during the transformation. A new restructuring operator is defined which minimizes that loss of information. Its definition is refined step by step into more and more computationally efficient versions.

1 Introduction

In 1977, Makinouchi [10] stressed that the first normal form condition imposed on relational databases was not convenient for handling a variety of database applications such as information retrieval systems or computer-aided design and manufacturing (CAD/CAM). These applications require the manipulation of structured entities while the 1NF condition only allows atomic values for attributes.

Nested relational databases are one attempt to meet these new requirements. They were first introduced in [8], where the 1NF condition is dropped and values of attributes are allowed to be sets. Two new operators, *nest* and *unnest* were defined for structuring and de-structuring nested relations. Those operators were then generalized so that values of attributes could themselves be nested relations [16], [2], [17], [15]. A nested relation is thus a set of tuples having themselves nested relations as values for some attributes. Various SQL-like languages were proposed for querying those databases [16], [4], [12], [13], [14].

A subclass of nested relations is particularly important for database applications. Those relations are sometimes said to be in *partitioned normal form* (PNF). A relation in PNF is such that its atomic attributes are a superkey of the relation and that any non-atomic component of a tuple of the relation is also in PNF. Relations in partitioned normal form have been studied in [3], [1], [15], and [6].

The present paper is concerned with the problem of restructuring PNF relations to new and potentially very different schemes. Section 2 first recalls some results from [2]: there is a one-to-one mapping from the class of PNF relations onto a well-defined subclass of the flat relational databases obeying the Universal Relation Schema Assumption. That mapping gives a precise semantics to PNF relations by associating each of them with a flat relational database. Next, an algorithmic description of that mapping and of its inverse is given. The problem of restructuring PNF relations then reduces to a simpler problem of restructuring flat relational databases. That approach eventually leads to the definition of a new restructuring operator which minimizes the amount of information lost during the transformation.

The operator defined in Section 2 is often inefficient from a computational point of view. Its definition is thus refined step by step in Section 3 into more and more computationally efficient versions.

Section 4 gives some conclusions. The related work of Abiteboul and Bidoit ([2], [1]) is discussed and some words are said on future works.

Basic concepts about nested relations and nested relational algebras are recalled in the rest of this section.

Basic concepts

Let U denote the universe of *atomic attributes*. Each atomic attribute A in U has a set of values associated with it, called the *domain* of A and denoted $dom(A)$.

Besides atomic attributes, there are *structured attributes*. A structured attribute X ($X \notin U$) is defined

Proceedings of the 16th VLDB Conference
Brisbane, Australia 1990

by an *attribute definition* of the form

$$X \equiv \{P_1, \ldots, P_n\}$$

where P_i $(1 \leq i \leq n)$ are atomic or structured attributes.

The following notational conventions for attributes are adopted:

- upper-case letters *A, B, C, D...* denote atomic attributes,
- upper-case letters *X, Y, Z...* denote structured attributes,
- overlined upper-case letters denote sets of attributes. The conventions above apply for differentiating sets of atomic and structured attributes. A set of attributes that may contain both atomic and structured attributes is represented by the overlined upper-case letters $\bar{P}$, $\bar{Q}$, or $\bar{R}$.

We also adopt the classical convention that unions of sets of attributes are denoted by concatenation. Then, an attribute definition defining a structured attribute Y is often written $Y \equiv \bar{A}\bar{X}$ where $\bar{A}$ and $\bar{X}$ are respectively its sets of atomic and structured attributes.

A *scheme* Δ is a union $\bar{R} \cup \mathcal{D}$ of a non-empty set $\bar{R}$ of atomic and structured attributes, called the *root* of the scheme, and of a set $\mathcal{D}$ of attribute definitions. A structured attribute X is *defined* in a relation scheme Δ if Δ contains a definition of the form $X \equiv \bar{P}$. A scheme Δ with root $\bar{R}$ is a *relation scheme* if the following conditions hold:

- any structured attribute that appears in the root of Δ or in the right-hand side of a definition of Δ is defined in Δ;
- no structured attribute is defined twice in Δ;
- no attribute appears in the right-hand side of several definitions or in the right-hand side of a definition and in the root of Δ.

A relation scheme is thus a hierarchical structure.

A useful notation is $rules(\bar{P}, \Delta)$ that denotes the set of rules recursively defining $\bar{P}$ in Δ. If $\bar{P} = \bar{A} \cup \{X_1 \ldots, X_n\}$, and if $X_i \equiv \bar{Q}_i$ are the definitions of X_i in Δ,

$$rules(\bar{P}, \Delta) = \{X_1 \equiv \bar{Q}_1, \ldots, X_n \equiv \bar{Q}_n\} \cup rules(\bar{Q}_1 \ldots \bar{Q}_n, \Delta).$$

If X is defined in Δ by $X \equiv \bar{P}$, the *sub-scheme* of Δ at X, denoted $sub(X, \Delta)$, is $\bar{P} \cup rules(\bar{P}, \Delta)$.

Let Δ be a relation scheme with root $\bar{A}\bar{X}$. A *nested relation* $\mathcal{R}$, or briefly a *relation*, with scheme Δ is a finite set of *tuples* over Δ, where a tuple t over Δ is a mapping of domain $\bar{A}\bar{X}$ such that $t[A] \in dom(A)$ for every $A \in \bar{A}$ and $t[X]$ is a relation (possibly empty) with scheme $sub(X, \Delta)$ for every $X \in \bar{X}$.

With the allowance of empty values for structured attributes, the modeling of null values of type "does not exist" is possible in nested relational databases.

The scheme of $\mathcal{R}$ is denoted $scheme(\mathcal{R})$ and, by extension, the *root* of $\mathcal{R}$ is the root of $scheme(\mathcal{R})$.

A *(nested) relational database* D is a set of (nested) relations. The *scheme* of D, denoted $Scheme(D)$, is the set of schemes of relations of D.

Observe that, if the set of attribute definitions of a relation scheme Δ is empty, then its root is a set $\bar{A}$ of atomic attributes. A relation with such a scheme is called *flat* or *1NF*. A *flat relational database* is a set of flat relations.

Example 1. To illustrate the concept of nested relation, consider a relation *Students* which represents the set of students in school X. Each student is modeled with two attributes: a student's name, *s_name* (atomic) and informations about the courses the student attends, *courses* (structured). The root of *Students* is thus {*s_name, courses*}. The attribute definitions for the structured attributes are:

courses $\equiv$ {*course, history*},
history $\equiv$ {*year, unit, teacher*},

where *course, year, unit, teacher* are atomic.

The structured attribute *history* recalls the set of units of a course that were already taken by a student in previous years and the teachers who taught the units.

A value for *Students* could be represented with the following nested table:

Students

s_name	*courses*			
	course	*history*		
		year	*unit*	*teacher*
Jones	*math*	1977	*A*	*Russell*
		1978	*B*	*Russell*
		1978	*C*	*Doolittle*
	science			
	physics	1977	*A*	*Martin*
		1978	*B*	*Anderson*
Smith	*physics*			

•

Definitions. Let $\mathcal{R}$ be a relation with root $\bar{R}$ and let $\bar{P} \subset \bar{R}$ and $\bar{Q} \subset \bar{R}$. The set of attributes $\bar{Q}$ *functionally depends* on the set of attributes $\bar{P}$ $(\bar{P} \rightarrow \bar{Q})$ if, for every tuples t_1 and t_2 in $\mathcal{R}$, if $t_1[\bar{P}] = t_2[\bar{P}]$, then $t_1[\bar{Q}] = t_2[\bar{Q}]$.

The relation $\mathcal{R}$ with scheme Δ and root $\bar{A}\bar{X}$ is in *partitioned normal form* (PNF) if

- $\bar{A} \neq \{\}$,
- $\bar{A} \rightarrow \bar{X}$,
- for every $t \in \mathcal{R}$ and every $X \in \bar{X}$, the sub-relation $t[X]$ is in PNF.

◇

Notice, in particular, that a flat relation is always in PNF. *Students* in Example 1 is in PNF.

The condition $\bar{A} \neq \{\}$ above is not mandatory but slightly simplifies our presentation.

A *PNF relational algebra* $\Sigma = \{\sqcup, \sqcap, -, \times, \pi, \rho, \sigma, \bowtie, \nu, \mu\}$ was defined in [6] for the class of PNF relations without empty attribute values. That algebra is the first one, using nesting and unnesting operators, under which the class of PNF relations is closed. The operators $\sqcap$, $-$, $\times$, ρ, σ, and $\bowtie$ of Σ are natural extensions of the operators $\cap$, $-$, $\times$, ρ, σ, and $\bowtie$ of the usual flat relational algebra. Their definitions are not recalled here. The definitions of $\sqcup$, π, ν, and μ are recalled hereafter.

Definitions.

Union operator $\sqcup$. Let $\mathcal{R}$ and $\mathcal{S}$ be PNF relations with scheme Δ and root $\bar{A}\bar{X}$. Then, $\mathcal{R} \sqcup \mathcal{S}$ is the set of tuples over Δ satisfying one of the following three conditions:
- $t \in \mathcal{R}$ and $t[\bar{A}] \neq t'[\bar{A}]$ for every $t' \in \mathcal{S}$;
- $t \in \mathcal{S}$ and $t[\bar{A}] \neq t'[\bar{A}]$ for every $t' \in \mathcal{R}$;
- $t[\bar{A}] = t_1[\bar{A}] = t_2[\bar{A}]$ for some $t_1 \in \mathcal{R}$ and $t_2 \in \mathcal{S}$ and $t[X] = t_1[X] \sqcup t_2[X]$ for every $X \in \bar{X}$.

Projection operator π. Let $\mathcal{R}$ be a PNF relation with scheme Δ and root $\bar{A}\bar{X}\bar{P}$. Then, $\pi_{\bar{A}\bar{X}}(\mathcal{R})$ is the set of tuples t over $\Delta' = \bar{A}\bar{X} \cup \mathit{rules}(\bar{X}, \Delta)$ such that, for some $u \in \mathcal{R}$ and for every X in $\bar{X}$,

$$t[\bar{A}] = u[\bar{A}],$$
$$t[X] = \bigsqcup\{v[X] \mid v \in \mathcal{R} \text{ and } v[\bar{A}] = u[\bar{A}]\}.$$

Nesting operator ν. Let $\mathcal{R}$ be a PNF relation with scheme Δ and root $\bar{A}\bar{X}\bar{Q}$. The *PNF nesting* of $\mathcal{R}$ along $Y \equiv \bar{Q}$ selects the sets of tuples of $\mathcal{R}$ equal on $\bar{A}$ and compacts them into tuples where the set of their different values on $\bar{Q}$ becomes values of the structured attribute Y and where the values on $\bar{X}$ are merged using union $\sqcup$. More precisely, if $\Delta' = \bar{A}\bar{X}Y \cup \mathit{rules}(\bar{X}, \Delta) \cup \{Y \equiv \bar{Q}\}$ is a relation scheme, then, $\nu_{Y \equiv \bar{Q}}(\mathcal{R})$ is the set of tuples t over Δ' such that, for some u in $\mathcal{R}$ and for every X in $\bar{X}$,

$$t[\bar{A}] = u[\bar{A}],$$
$$t[X] = \bigsqcup\{v[X] \mid v \in \mathcal{R} \text{ and } v[\bar{A}] = u[\bar{A}]\},$$
$$t[Y] = \{v[\bar{Q}] \mid v \in \mathcal{R} \text{ and } v[\bar{A}] = u[\bar{A}]\}.$$

Unnesting operator μ. Let Δ be a relation scheme with root $\bar{P}Y$, $Y \equiv \bar{Q}$ be the definition of Y in Δ and $\mathcal{R}$ be a PNF relation with scheme Δ. The *unnesting* of $\mathcal{R}$ along $Y \equiv \bar{Q}$, denoted $\mu_Y(\mathcal{R})$, disaggregates the structured attribute Y. More precisely, $\mu_Y(\mathcal{R})$ is the set of tuples t over $\Delta' = \bar{P}\bar{Q} \cup \mathit{rules}(\bar{P}\bar{Q}, \Delta)$ such that for some tuple u in $\mathcal{R}$,

$$t[\bar{P}] = u[\bar{P}] \text{ and}$$
$$t[\bar{Q}] \in u[Y].$$

◇

The unnesting operator is the usual one. The union and projection operators $\sqcup$ and π are identical to the extended union and projection operators of [15]. The nesting operator ν has been introduced in [6] and differs from the usual nesting operator ν which is, with the above notations, the set of tuples t over Δ' such that, for some u in $\mathcal{R}$,

$$t[\bar{A}\bar{X}] = u[\bar{A}\bar{X}],$$
$$t[Y] = \{v[\bar{Q}] \mid v \in \mathcal{R} \text{ and } v[\bar{A}] = u[\bar{A}]\}.$$

We proved several important properties of ν in [6].

- *If $\mathcal{R}$ is a PNF relation with root $\bar{A}\bar{X}\bar{Q}$, then, $\nu_{Y \equiv \bar{Q}}(\mathcal{R})$ is a PNF relation.* That property is not true for ν in general.
- *Let $\mathcal{R}$ be a PNF relation. If $\nu_{Y \equiv \bar{Q}}(\mathcal{R})$ is in PNF, then $\nu_{Y \equiv \bar{Q}}(\mathcal{R}) = \nu_{Y \equiv \bar{Q}}(\mathcal{R})$.* That property proves that ν is quite close to ν. They only differ when ν maps PNF relations to non-PNF relations.
- *Let $\mathcal{R}$ be a PNF relation with root $\bar{P}\bar{Q}\bar{R}$. Then, $\nu_{W_1 = \bar{Q}}\nu_{W_2 = \bar{R}}(\mathcal{R}) = \nu_{W_2 = \bar{R}}\nu_{W_1 = \bar{Q}}(\mathcal{R})$.* That property of commutativity is very important for optimization of relational expressions and is not true for ν in general.
- The computation of $\nu_{Y \equiv \bar{Q}}(\mathcal{R})$ often needs comparisons of sets for equality while $\nu_{Y \equiv \bar{Q}}(\mathcal{R})$ only needs to form their union. Thus, $\nu_{Y \equiv \bar{Q}}(\mathcal{R})$ is more quickly computed than $\nu_{Y \equiv \bar{Q}}(\mathcal{R})$.

We also argued in [6] that a non-PNF relation $\mathcal{R}'$ obtained from a PNF relation $\mathcal{R}$ by nestings with ν often has no simple semantic interpretation contrary to the PNF relation $\mathcal{R}''$ obtained by similar nestings with ν.

2 Restructuring of PNF relations

2.1 General considerations

The aim of this paper is to study restructuring in the class of PNF relations. Let P_Δ denote the set of PNF relations with scheme Δ. A *restructuring operator* R_Δ

is an operator that maps the set of PNF relations to P_Δ.

Example 2. Consider Relation *Students* of Example 1 and the new relation scheme Δ',

$\{s\_name, curriculum,$
$curriculum \equiv \{year, courses\},$
$courses \equiv \{course, units, teachers\},$
$units \equiv \{unit\},$
$teachers \equiv \{teacher\}$ $\}$.

The restructured relation $R_{\Delta'}(Students)$ must then be interpreted as the set of students in school X together with their curriculum. The curriculum associates with each year that a student spent in X, the set of courses he attended. Each course is characterized by its name, by the list of units attended that year by the student, and by the set of teachers for those units.

With that interpretation, the restructured relation $R_{\Delta'}(Students)$ should be

$R_{\Delta'}(Students)$

s_name	curriculum			
	year	courses		
		course	units	teachers
			unit	teacher
Jones	1977	math	A	Russell
		physics	A	Martin
	1978	math	B	Russell
			C	Doolittle
		physics	B	Anderson
Smith				

•

Before giving a precise definition to R_Δ, we want to stress that restructuring often entails some loss of information. The loss of information can have two origins as the following examples show.

Example 3. Consider the following relation $\mathcal{R}_1$.

$\mathcal{R}_1$

dpt	part	sub-parts	
		sub-part	supplier
D1	p1	sp1	s1
		sp2	s2
		sp3	s1
D2	p2		

Relation $\mathcal{R}_1$ means that department $D1$ uses subparts $sp1$, $sp2$, $sp3$ from suppliers $s1$, $s2$, and $s1$ respectively in the manufacturing of part $p1$. Department $D2$ manufactures part $p2$ without using subparts from outside.

Relation $\mathcal{R}_1$ can be restructured to the flat scheme with root $\{dpt, part, sub\text{-}part, supplier\}$. The resulting relation is $\mathcal{R}_2 = \mu_{sub\text{-}part}(\mathcal{R}_1)$.

$\mathcal{R}_2$

dpt	part	sub-part	supplier
D1	p1	sp1	s1
D1	p1	sp2	s2
D1	p1	sp3	s1

After the flattening of $\mathcal{R}_1$, the information concerning department $D2$ is lost. •

That example shows that the presence of empty values for structured attributes is a cause for information loss in restructuring. That is not the only one as the following example shows.

Example 4. The restructuring of Relation $\mathcal{R}_2$ of Example 3 to the scheme Δ'

$\{dpt, part, sub\text{-}parts, suppliers,$
$sub\text{-}parts \equiv sub\text{-}part,$
$suppliers \equiv supplier$ $\}$

associates, with each department and each part manufactured in that department, the set of sub-parts used during the manufacturing process and the set of suppliers for these parts.

$\mathcal{R}_3$

dpt	part	sub-parts	suppliers
		sub-part	supplier
D1	p1	sp1	s1
		sp2	s2
		sp3	

The restructured relation $\mathcal{R}_3$ is defined in terms of $\mathcal{R}_2$ by the relational expression $\nu_{sub\text{-}parts \equiv sub\text{-}part}\nu_{suppliers \equiv supplier}(\mathcal{R}_2)$. Information is lost again during restructuring: it is not known any more which supplier supplies which part. •

The above example shows that, when a relation with root $\bar{P}\bar{Q}_1\bar{Q}_2$ is successively nested along $\bar{Q}_1$ and $\bar{Q}_2$, the associations between values of $\bar{Q}_1$ and $\bar{Q}_2$ that existed for every value of $\bar{P}$ are lost.

These losses of information are normal. The semantics of a restructured relation is different from that of the original relation and the new semantics cannot always capture the whole information conveyed by the old semantics.

Examples 3 and 4 constitute also a good basis for a very important remark. The nesting ν and unnesting μ operators perform simple restructurings and, if Δ and Δ' are built upon the same set of atomic attributes, there are clearly an infinite number of compositions of these two operators that restructure a relation $\mathcal{R}$ with scheme Δ to scheme Δ'. It could be believed that one among them would minimize the loss of information. This is unfortunately not true as the following example shows.

Example 5. Consider the restructuring of Relation $\mathcal{R}_1$ of Example 3 directly to Scheme Δ' of Example 4. The loss of the information between suppliers and the parts they supply is inherent to the restructuring. The loss of the information about D_2 is not. The resulting relation $\mathcal{R}_4$ should be

$\mathcal{R}_4$

dpt	*part*	*sub-parts*	*suppliers*
		sub-part	*supplier*
D1	*p1*	*sp1* *sp2* *sp3*	*s1* *s2*
D2	*p2*		

Relation $\mathcal{R}_4$ is different from the composition $\nu_{sub\text{-}parts \equiv sub\text{-}part}\nu_{suppliers \equiv supplier}\mu_{sub\text{-}parts}(\mathcal{R}_1)$ of restructurings of Examples 3 and 4. More generally, every composition of ν and μ that restructures $\mathcal{R}_1$ to scheme Δ' must perform an unnesting along *sub-parts* and causes the loss of the information on department $D2$. •

That example shows that nesting and unnesting are not an adequate basis for performing restructurings when structured attributes admit empty values. It demonstrates the need for more general restructuring operators that guarantee the minimization of the loss of information.

2.2 Flat interpretation of PNF relations

Let $\mathcal{R}'$ in $\boldsymbol{P}_{\Delta'}$ be the restructuring of $\mathcal{R}$ in $\boldsymbol{P}_{\Delta}$. The structure of Δ' can be radically different from that of Δ and this makes a direct definition of $R_{\Delta'}$ difficult. Therefore, flat equivalent representations are associated with PNF relations. Restructurings are then more easily defined via those equivalent representations.

It appears from Example 3 that a flat equivalent representation cannot simply be obtained by complete unnesting since information is lost during this process. Two operators Φ_Δ and Σ_Δ are defined in the following such that

- for every $\mathcal{R}$ in $\boldsymbol{P}_\Delta$, $\Phi_\Delta(\mathcal{R})$ is a set of flat relations,
- for every $\mathcal{R}$ in $\boldsymbol{P}_\Delta$, $\Sigma_\Delta(\Phi_\Delta(\mathcal{R})) = \mathcal{R}$.

Since Φ_Δ is invertible, $\Phi_\Delta(\mathcal{R})$ is a set of flat relations "equivalent" to $\mathcal{R}$.

Example 6. Consider again Relation *Students* of Example 1. Similarly to Example 3, the presence of empty values in *Students* leads to information loss after flattening through unnesting. That loss can be suppressed if three relations instead of one result from the flattening process:

- the first relation $\mathcal{S}_1$ has one attribute *s_name* and remembers the set of students,
- the second relation $\mathcal{S}_2$ has two attributes *s_name* and *course* and remembers the students who attend courses and these courses,
- the third relation has five attributes *s_name*, *course*, *year*, *unit*, and *grade* and remembers the students who have results in some units of courses, these courses, the units, and the teachers who taught them.

The three relations are given hereafter. They constitute the flat equivalent of relation *Students* in the sense that they can be built from *Students* and that, conversely, *Students* can be re-built from them.

$\mathcal{S}_1$

s_name
Jones
Smith

$\mathcal{S}_2$

s_name	*course*
Jones	*math*
Jones	*science*
Jones	*physics*
Smith	*physics*

$\mathcal{S}_3$

s_name	*course*	*year*	*unit*	*teacher*
Jones	*math*	1977	*A*	*Russel*
Jones	*math*	1978	*B*	*Russel*
Jones	*math*	1978	*C*	*Doolittle*
Jones	*physics*	1977	*A*	*Martin*
Jones	*physics*	1978	*B*	*Anderson*

•

The following definitions formalize this new flattening process. The flattening of relations of $\boldsymbol{P}_\Delta$ always results in flat relational databases with the same scheme. The mapping φ induced by the flattening between PNF relations schemes and flat database schemes can be defined as follows.

Definitions. If $\mathcal{R}$ is a relation in $\boldsymbol{P}_\Delta$ with root $\bar{A}\bar{X}$, then, for every X in $\bar{X}$, Δ_X is the scheme of $\mu_X \pi_{\bar{A}X}(\mathcal{R})$.

$$\Delta_X = \bar{A} \cup sub(X, \Delta).$$

If Δ is a relation scheme with root $\bar{A}\bar{X}$, then,

$$\varphi(\Delta) = \{\bar{A}\} \cup \bigcup_{X \in \bar{X}} \varphi(\Delta_X). \quad \diamond$$

If Δ' is the scheme of Relation $\mathcal{R}_4$ in Example 5, $\varphi(\Delta') = \{\{dpt, part\}, \{dpt, part, sub\text{-}part\}, \{dpt, part, supplier\}\}$.

Definition. The *flattening operator* Φ_Δ maps PNF relations with scheme Δ to flat relational databases with scheme $\varphi(\Delta)$. Let $\mathcal{R}$ be a PNF relation with root $\bar{A}\bar{X}$. Then,

$$\Phi_\Delta(\mathcal{R}) = \{\pi_{\bar{A}}(\mathcal{R})\} \cup \bigcup_{X \in \bar{X}} \Phi_{\Delta_X}(\mu_X \pi_{\bar{A}X}(\mathcal{R})). \quad \diamond$$

It can be easily verified that, in Example 6, $\Phi_\Delta(Students) = \{\mathcal{S}_1, \mathcal{S}_2, \mathcal{S}_3\}$.

From now on, the *flattening* of a relation $\mathcal{R}$ will be the flat relational database $\Phi_\Delta(\mathcal{R})$. In order to characterize the class of flat relational databases which are flattenings of PNF relations, some new definitions are introduced.

Definitions. An *URSA database* D is a flat relational database that satisfies the *Universal Relation Schema Assumption* (URSA), i.e., such that $\pi_{scheme(\mathcal{R}_1)}(\mathcal{R}_2) \subset \mathcal{R}_1$ for every relation $\mathcal{R}_1$ and $\mathcal{R}_2$ of D with $scheme(\mathcal{R}_1) \subset scheme(\mathcal{R}_2)$ [9].

A database scheme $\mathcal{S}$ is *closed for intersection* if, for all schemes S_1 and S_2 in $\mathcal{S}$, there exists a scheme S_3 in $\mathcal{S}$ such that $S_3 = S_1 \cap S_2$.

A database scheme $\mathcal{S}$ is *hierarchical* if

- $\mathcal{S}$ is closed for intersection,
- for each S in $\mathcal{S}$, $\{S \cap T \mid T \in \mathcal{S}\}$ is totally ordered by inclusion.

$\diamond$

Let $\boldsymbol{D_S}$ denote the set of URSA databases with hierarchical scheme $\mathcal{S}$.

The following theorem, adapted from [1], exactly characterizes the class of flat relational databases that are flattenings of PNF relations.

Theorem 1. *If Δ is a relation scheme and if $\mathcal{S} = \varphi(\Delta)$, then, Φ_Δ is a one-to-one mapping from $\boldsymbol{P_\Delta}$ onto $\boldsymbol{D_S}$.*

That theorem shows that flattenings losslessly represent PNF relations.

An algebraic definition of the inverse Σ_Δ of Φ_Δ is needed in the following. That definition uses an operator ν_Δ.

Definition. Let Δ be a relation scheme with root $\bar{A}X_1 \ldots X_n$ and let $X_i \equiv \bar{Q}_i$ be the attribute definitions of X_i in Δ $(1 \leq i \leq n)$. Then, ν_Δ takes two arguments:

- a flat relation $\mathcal{R}$ with scheme $\bar{A}$,
- a set of nested relations $\mathcal{R}_1, \ldots, \mathcal{R}_n$ with schemes $\Delta_{X_1}, \ldots, \Delta_{X_n}$ respectively.

Then,

$$\begin{aligned}\nu_\Delta(\mathcal{R}, \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}) \\ = \{t \mid &\text{for some } u \in \mathcal{R}, \text{ for every } i \in \{1, \ldots, n\}, \\ &t[\bar{A}] = u, \\ &t[X_i] = \{v[\bar{Q}_i] \mid v \in \mathcal{R}_i \text{ and } v[\bar{A}] = u\}\}.\end{aligned}$$ $\diamond$

For every tuple u in $\mathcal{R}$, $\nu_\Delta(\mathcal{R}, \{\mathcal{R}_1, \ldots, \mathcal{R}_n\})$ generates exactly one tuple t such that $t[\bar{A}] = u$. In particular, if $u \notin \pi_{\bar{A}}(\mathcal{R}_i)$ for some i in $\{1, \ldots, n\}$, then $t[X_i] = \{\}$. The definition of Σ_Δ can now be given.

Definition. If Δ is a relation scheme with root $\bar{A}\bar{X}$, and if D is a database in $\boldsymbol{D_S}$ such that $\mathcal{S}$ contains $\varphi(\Delta)$,

$$\Sigma_\Delta(D) = \nu_\Delta(rel(\bar{A}, D), \{\Sigma_{\Delta_X}(D) \mid X \in \bar{X}\}),$$

where $rel(\bar{A}, D)$ is the relation of D with scheme $\bar{A}$. $\diamond$

It is very easy to verify that the definition is well formed since D indeed contains a relation with scheme $\bar{A}$ and since $\varphi(\Delta_X)$ is contained in $\mathcal{S}$ for every $X \in \bar{X}$. The following theorem shows that Σ_Δ is the inverse of Φ_Δ[1].

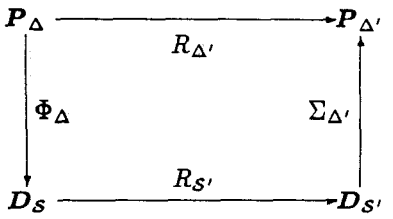
Theorem 2. *Σ_Δ is the inverse of Φ_Δ, that is, for every relation $\mathcal{R}$ with scheme Δ,*

$$\Sigma_\Delta(\Phi_\Delta(\mathcal{R})) = \mathcal{R}.$$

2.3 Restructuring operators

The problem of defining restructuring operators R_Δ can now be reduced to a problem of restructuring flat URSA databases with hierarchical schemes as the picture hereafter shows.

[1] Proofs of theorems cited in this paper can be found in [5].



If $\varphi(\Delta) = \mathcal{S}$ and if $\varphi(\Delta') = \mathcal{S}'$, the problem reduces to defining a restructuring operator $R_{\mathcal{S}'}$ from $D_\mathcal{S}$ to $D_{\mathcal{S}'}$.

An URSA database never contains two relations over the same scheme, for both would contain the same set of tuples. In an URSA database, the scheme is sufficient to identify a relation. More generally, if $scheme(\mathcal{R}_1) \subset scheme(\mathcal{R}_2)$, then, $\pi_{scheme(\mathcal{R}_1)}(\mathcal{R}_2) \subset \mathcal{R}_1$. The URSA on a database scheme can thus be interpreted as requiring that an attribute mean the same everywhere it appears. A consequence of the URSA is that tuples over a given set of attributes have a single meaning. A function $[\bar{A}, D]$ that associates with each URSA database D and set of attributes $\bar{A}$ a relation with scheme $\bar{A}$ is called a *window function* [9].

Let D be an URSA database with hierarchical scheme $\mathcal{S}$. A relation of D, denoted $[A, D]_1$, that collects the largest amount of information contained in D on A, is associated with each attribute A.

If A is attribute of no relation of D, then there is no information available on A in D and $[A, D]_1 = \{\}$. Otherwise, let $\mathcal{R}_1, \ldots, \mathcal{R}_n$ be the relations of D such that $A \in scheme(\mathcal{R}_i)$ $(1 \leq i \leq n)$. As $\mathcal{S}$ is closed for intersection, there exists some j $(1 \leq j \leq n)$ such that $scheme(\mathcal{R}_j) \subset scheme(\mathcal{R}_i)$ for every $i = 1, \ldots, n$. It then follows, by the URSA on D, that $\mathcal{R}_j \supset \pi_{scheme(\mathcal{R}_j)}(\mathcal{R}_i)$ for every $i = 1, \ldots, n$. The relation $\mathcal{R}_j$ is thus the most informative relation in D on attribute A and $[A, D]_1 = \mathcal{R}_j$.

The information contained in D on a set of attributes $\bar{A}$ is then obtained by joining the relations associated with each attribute of $\bar{A}$ and projecting on $\bar{A}$:

$$[\bar{A}, D] = \pi_{\bar{A}}(\bowtie_{B \in \bar{A}} [B, D]_1).$$

The restructuring operator $R_\mathcal{S}$, has now a very natural definition:

$$R_\mathcal{S}(D) = \{[\bar{A}, D] \mid \bar{A} \in \mathcal{S}\}.$$

We can now give the definition of the restructuring operator R_Δ.

Definition. If Δ' is a relation scheme and if $\varphi(\Delta') = \mathcal{S}'$, then,

$$R_\Delta(\mathcal{R}) = \Sigma_{\Delta'}(R_{\mathcal{S}'}(\Phi_\Delta(\mathcal{R}))). \qquad \diamond$$

Example 7. Consider the relation *Students* of Example 1 and its flattening in Example 6. If Δ' is the relation scheme of Example 2 and if $\mathcal{S}' = \varphi(\Delta')$, $R_{\mathcal{S}'}(\Phi_\Delta(Students))$ contains five relations $\mathcal{S}'_1$, $\mathcal{S}'_2$, $\mathcal{S}'_3$, $\mathcal{S}'_4$, and $\mathcal{S}'_5$, with schemes {*s_name*}, {*s_name, year*}, {*s_name, year, course*}, {*s_name, year, course, unit*}, and {*s_name, year, course, teacher*} respectively. These relations are

$\mathcal{S}'_1$

s_name
Jones
Smith

$\mathcal{S}'_2$

s_name	*year*
Jones	1977
Jones	1978

$\mathcal{S}'_3$

s_name	*year*	*course*
Jones	1977	*math*
Jones	1977	*physics*
Jones	1978	*math*
Jones	1978	*physics*

$\mathcal{S}'_4$

s_name	*year*	*course*	*unit*
Jones	1977	*math*	*A*
Jones	1977	*physics*	*A*
Jones	1978	*math*	*B*
Jones	1978	*math*	*C*
Jones	1978	*physics*	*B*

$\mathcal{S}'_5$

s_name	*year*	*course*	*teacher*
Jones	1977	*math*	*Russel*
Jones	1977	*physics*	*Martin*
Jones	1978	*math*	*Russel*
Jones	1978	*math*	*Doolittle*
Jones	1978	*physics*	*Anderson*

It is easily verified that $\Sigma_{\Delta'}(R_{\mathcal{S}'}(\Phi_\Delta(Students)))$ is the nested relation of Example 2. •

The definition of $R_{\Delta'}$ can be transformed. Let $\mathcal{R}$ be a relation with scheme Δ, and let Δ' be a relation scheme with root $\bar{A}'\bar{X}'$. If $R^t_{\Delta'} = \Sigma_{\Delta'}.R_{\mathcal{S}'}$, then,

$$R_{\Delta'}(\mathcal{R}) = R^t_{\Delta'}(\Phi_\Delta(\mathcal{R}))$$

and, by induction on Δ',

$$\begin{aligned} &R^t_{\Delta'}(\Phi_\Delta(\mathcal{R})) \\ &\quad = \Sigma_{\Delta'}(R_{\varphi(\Delta')}(\Phi_\Delta(\mathcal{R}))) \\ &\quad = \nu_{\Delta'}(rel(\bar{A}', R_{\varphi(\Delta')}(\Phi_\Delta(\mathcal{R}))), \\ &\qquad\quad \{\Sigma_{\Delta'_{X'}}(R_{\varphi(\Delta')}(\Phi_\Delta(\mathcal{R}))) \mid X' \in \bar{X}'\}) \\ &\quad = \nu_{\Delta'}([\bar{A}', \Phi_\Delta(\mathcal{R})], \{R^t_{\Delta'_{X'}}(\Phi_\Delta(\mathcal{R})) \mid X' \in \bar{X}'\}). \end{aligned}$$

Indeed, $R_{\varphi(\Delta'_{X'})}(\Phi_\Delta(\mathcal{R})) \subset R_{\varphi(\Delta')}(\Phi_\Delta(\mathcal{R}))$ by definition and $\Sigma_{\Delta'_{X'}}(D)$ only uses relations of D with schemes in $\varphi(\Delta'_{X'})$. It follows that

$\Sigma_{\Delta'_{X'}}(R_{\varphi(\Delta')}(\Phi_\Delta(\mathcal{R}))) = \Sigma_{\Delta'_{X'}}(R_{\varphi(\Delta'_{X'})}(\Phi_\Delta(\mathcal{R}))) = R_{\Delta'_{X'}}(\mathcal{R}) = R^t_{\Delta'_{X'}}(\Phi_\Delta(\mathcal{R}))$.

In summary, the restructuring operator R is defined as shown in Figure 1.

If $\mathcal{R}$ is a PNF relation with scheme Δ and root $\bar{A}\bar{X}$ and if Δ' is a relation scheme with root $\bar{A}'\bar{X}'$,

$$R_{\Delta'}(\mathcal{R}) = R^t_{\Delta'}(\Phi_\Delta(\mathcal{R}))$$

$$R^t_{\Delta'}(D) = \nu_{\Delta'}([\bar{A}', D], \{R^t_{\Delta'_{X'}}(D) \mid X' \in \bar{X}'\}),$$

$$\Phi_\Delta(\mathcal{R}) = \{\pi_{\bar{A}}(\mathcal{R})\} \cup \bigcup_{X \in \bar{X}} \Phi_{\Delta_X}(\mu_X \pi_{\bar{A}X}(\mathcal{R})),$$

where, for every set of atomic attributes $\bar{B}$ and every URSA database D with hierarchical scheme,

$$[\bar{B}, D] = \pi_{\bar{B}}(\bowtie_{B \in \bar{B}} [B, D]_1),$$

and where, for every relation scheme Δ with root $\bar{A}X_1 \ldots X_n$ and for every relations $\mathcal{R}, \mathcal{R}_1, \ldots, \mathcal{R}_n$ with schemes $\{\bar{A}\}, \Delta_{X_1}, \ldots, \Delta_{X_n}$ respectively, if X_i are defined in Δ by $X_i \equiv \bar{Q}_i$ $(1 \leq i \leq n)$,

$$\nu_\Delta(\mathcal{R}, \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}) = \{t \mid \text{for some } u \in \mathcal{R} \text{ and for every } i \in \{1, \ldots, n\}, \; t[\bar{A}] = u, \; t[X_i] = \{v[\bar{Q}_i] \mid v \in \mathcal{R}_i \text{ and } v[\bar{A}] = u\} \}.$$

Figure 1: Restructuring operator – Version I

3 Optimizations

The above definition of $R_{\Delta'}$ can be optimized. Indeed, consider two relation schemes Δ and Δ' and the restructuring $R_{\Delta'}(\mathcal{R})$ of a relation $\mathcal{R}$ of P_Δ. It often happens that Δ and Δ' share isomorphic sub-schemes.

Example 8. Consider again Relation *Students* of Example 1 and the scheme Δ'
$\{course, students,$
$students \equiv \{student, curriculum\},$
$curriculum \equiv \{year, unit, teacher\}\}$.

The attributes *history* in Δ and *curriculum* in Δ' clearly have isomorphic structures. •

The problem is then to optimize the definition of $R_{\Delta'}$ so that sub-schemes of Δ which are isomorphic to sub-schemes of Δ' are not flattened. That optimization is presented in Section 3.1.

Now consider two relations $\mathcal{R}$ and $\mathcal{R}'$ with schemes Δ and Δ' depicted as follows.

Δ

name	*year*	*sports*				
		sport	*trainer*	*competitions*		
				date	*place*	*comment*

Δ'

name	*sports*			
	sport	*trainer*	*competitions*	
			place	*results*
				date	*result*

Attributes *competitions* in Δ and Δ' are not isomorphic but, as

- no atomic attributes in $sub(competitions, \Delta')$ is an atomic attribute in $\Delta \setminus sub(competitions, \Delta)$,
- no atomic attribute in $\Delta \setminus sub(competitions, \Delta)$ is an atomic attribute in $sub(competitions, \Delta')$,

all sub-relations of $\mathcal{R}$ with scheme $sub(competitions, \Delta)$ could be restructured to scheme $sub(competitions, \Delta')$ independently of the values that the tuples of $\mathcal{R}$ where the sub-relations are nested take on $\Delta \setminus sub(competitions, \Delta)$.

This leads to an optimization where restructurings are performed locally on sub-relations whenever possible rather then on the whole relation. Efficiency is thus increased as window functions for local restructurings on sub-relations are performed by joining fewer relations with fewer attributes. That optimization is presented in Section 3.2.

3.1 Isomorphic attributes

This section presents an optimized version $R^{iso}_{\Delta'}(\mathcal{R})$ of $R_{\Delta'}(\mathcal{R})$ such that sub-relations of $\mathcal{R}$ with schemes isomorphic to sub-schemes of Δ' are not flattened. First, some definitions and notations about isomorphism between attributes are introduced.

Definitions. Two relation schemes Δ_1 and Δ_2 are *isomorphic* if they are equal up to a renaming of their structured attributes.

Let X_1 and X_2 be two structured attributes defined in relation schemes Δ_1 and Δ_2 respectively. Then, X_1 in Δ_1 and X_2 in Δ_2 are *isomorphic* if $sub(X_1, \Delta_1)$ is isomorphic to $sub(X_2, \Delta_2)$. They are *maximally isomorphic* if there are no isomorphic attributes Y_1 in Δ_1 and Y_2 in Δ_2 such that X_1 and X_2 are defined in $sub(Y_1, \Delta_1)$ and $sub(Y_2, \Delta_2)$ respectively.

The expression $iso(\Delta_1, \Delta_2)$ denotes the set of structured attributes defined in Δ_1 which are maximally isomorphic to attributes in Δ_2. ◇

Consider now a relation $\mathcal{R}$ with scheme Δ to be restructured to scheme Δ'. The definition of the flat-

tening operator Φ_Δ is modified in order not to flatten attributes in Δ that are maximally isomorphic to attributes in Δ'.

Definition. Let $\mathcal{R}$ be a PNF relation with scheme Δ and root $\bar{A}\bar{X}$. Let $\bar{X}_1 = \bar{X} \cap iso(\Delta, \Delta')$ and $\bar{X}_2 = \bar{X} \setminus iso(\Delta, \Delta')$. Then, if $\rho_{\Delta \to \Delta'}$ denotes an operator that renames X to X' for every pair of isomorphic attributes X in Δ and X' in Δ',

$$\Phi^{iso}_\Delta(\mathcal{R}, \Delta') = \{\rho_{\Delta \to \Delta'} \pi_{\bar{A}\bar{X}_1}(\mathcal{R})\} \cup \bigcup_{X \in \bar{X}_2} \Phi^{iso}_{\Delta_X}(\mu_X \pi_{\bar{A}X}(\mathcal{R}), \Delta').$$

◇

The window functions must also be generalized. The definitions of $[\_,\_]_1$ and $[\_,\_]$ naturally extend to allow structured attributes and nested databases in their arguments but this is not enough as the following example shows.

Example 9. Consider the following very simple relation $\mathcal{R}$

$\mathcal{R}$

A	X	
	B	Z
		C
a		

and let $\Delta' = \{A, Z, Z \equiv C\}$.

Then, $\Phi^{iso}_\Delta(\mathcal{R}, \Delta')$ contains relations $\mathcal{R}_1$ with scheme $\{A\}$ and $\mathcal{R}_2$ with scheme $\{A, B, Z, Z \equiv C\}$. Relation $\mathcal{R}_1$ contains a and relation $\mathcal{R}_2$ is empty. It follows that

$$\begin{aligned}[AZ, \Phi^{iso}_\Delta(\mathcal{R}, \Delta')] &= \pi_{AZ}([A, \Phi^{iso}_\Delta(\mathcal{R}, \Delta')]_1 \bowtie [Z, \Phi^{iso}_\Delta(\mathcal{R}, \Delta')]_1) \\ &= \pi_{AZ}(\mathcal{R}_1 \bowtie \mathcal{R}_2) \\ &= \{\}\end{aligned}$$

This is clearly unsatisfactory as the expected answer is of course $\{<a, \{\}>\}$. •

A slightly generalized window function $[\_,\_]^{iso}$ is thus defined. A tuple u is built in $[\bar{A}\bar{X}, D]^{iso}$ for every tuple v in $[\bar{A}, D]$. Its value $u[\bar{A}]$ on $\bar{A}$ is v. For every X in $\bar{X}$, $[\bar{A}X, D]$ is in PNF and therefore contains zero or one tuple w such that $w[\bar{A}] = v$. Then, $u[X] = \{\}$ or $u[X] = w[X]$ respectively.

A formal definition of $[\_,\_]^{iso}$ that uses ν_Δ is now given.

Definition. Let $\mathcal{R}$ be a PNF relation with scheme Δ and let $D = \Phi^{iso}_\Delta(\mathcal{R}, \Delta')$ for some scheme Δ'. Then, if $\bar{A}$ is a set of atomic attributes and if $\bar{X}$ is a subset of $iso(\Delta', \Delta)$,

$$[\bar{A}\bar{X}, D]^{iso} = \nu_{\Delta'(\bar{A}\bar{X})}([\bar{A}, D], \{\mu_X([\bar{A}X, D]) \mid X \in \bar{X}\}),$$

where $\Delta'(\bar{A}\bar{X}) = \{\bar{A}\bar{X}\} \cup rules(\bar{X}, \Delta')$. ◇

With that new definition, $[AZ, \Phi^{iso}_\Delta(\mathcal{R}, \Delta')]^{iso} = \{<a, \{\}>\}$ in Example 9.

The optimized restructuring operator R^{iso} is then defined as follows.

Definition. Let Δ and Δ' be relation schemes. If $\bar{A}\bar{X}$ is the root of Δ' and if $\bar{X}_1 = \bar{X} \cap iso(\Delta', \Delta)$ and $\bar{X}_2 = \bar{X} \setminus iso(\Delta', \Delta)$,

$$R^{iso}_{\Delta'}(\mathcal{R}) = R^{i}_{\Delta \to \Delta'}(\Phi^{iso}_\Delta(\mathcal{R}, \Delta')),$$

$$R^{i}_{\Delta \to \Delta'}(D) = \nu_{\Delta'}([\bar{A}\bar{X}_1, D]^{iso}, \{R^{i}_{\Delta \to \Delta'_{X_2}}(D) \mid X_2 \in \bar{X}_2\}),$$

where, for every relation scheme Δ with root $\bar{A}\bar{Y}X_1 \ldots X_n$ and for every relations $\mathcal{R}, \mathcal{R}_1, \ldots, \mathcal{R}_n$ with schemes $\Delta(\bar{A}\bar{Y}), \Delta_{X_1}, \ldots, \Delta_{X_n}$ respectively, if X_i are defined in Δ by $X_i \equiv \bar{Q}_i$ $(1 \leq i \leq n)$,

$$\begin{aligned}\nu_\Delta(\mathcal{R}, \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}) = \{t \mid &\text{ for some } u \in \mathcal{R}, \text{ for every } i \in \{1, \ldots, n\}, \\ &t[\bar{A}\bar{Y}] = u, \\ &t[X_i] = \{v[\bar{Q}_i] \mid v \in \mathcal{R}_i \text{ and } v[\bar{A}] = u[\bar{A}]\}\}.\end{aligned}$$

◇

The following theorem shows that the restructuring operators R and R^{iso} are equivalent.

Theorem 3. *For every relation scheme Δ' and every PNF relation $\mathcal{R}$,*

$$R_{\Delta'}(\mathcal{R}) = R^{iso}_{\Delta'}(\mathcal{R}).$$

The new version of the restructuring operator after optimization is summarized in Figure 2.

3.2 Restructurable attributes

The second optimization aims to perform restructuring locally on sub-relations rather than globally on the whole relation. If $\mathcal{R}$ with scheme Δ is restructured to Δ', pairs of structured attributes (X, X') are found, such that sub-relations of $\mathcal{R}$ with scheme $sub(X, \Delta)$ can be safely restructured to scheme $sub(X', \Delta')$. Attribute X is then said to be *restructurable* to attribute X'.

Pairs (X, X') must satisfy the following condition: every atomic attribute of $sub(X, \Delta)$ (resp. $sub(X', \Delta')$) which is attribute of Δ' (resp. Δ) is at-

Let $\mathcal{R}$ be in $\boldsymbol{P}_\Delta$ with root $\bar{A}\bar{X}$
Δ' be a relation scheme with root $\bar{A}'\bar{X}'$,
$\bar{X}_1 = \bar{X} \cap iso(\Delta, \Delta')$ and $\bar{X}_2 = \bar{X} \setminus iso(\Delta, \Delta')$,
$\bar{X}'_1 = \bar{X}' \cap iso(\Delta', \Delta)$ and $\bar{X}'_2 = \bar{X}' \setminus iso(\Delta', \Delta)$,

$$R^{iso}_{\Delta'}(\mathcal{R}) = R^{i}_{\Delta\to\Delta'}(\Phi^{iso}_{\Delta}(\mathcal{R}, \Delta')),$$

$$R^{i}_{\Delta\to\Delta'}(D) = \nu_{\Delta'}([\bar{A}'\bar{X}'_1, D]^{iso}, \{R^{i}_{\Delta\to\Delta'_{X'_2}}(D) \mid X'_2 \in \bar{X}'_2\}),$$

$$\Phi_\Delta(\mathcal{R}, \Delta')^{iso} = \{\rho_{\Delta\to\Delta'}\pi_{\bar{A}\bar{X}_1}(\mathcal{R})\} \cup \bigcup_{X\in\bar{X}_2} \Phi^{iso}_{\Delta_X}(\mu_X\pi_{\bar{A}X}(\mathcal{R}), \Delta'),$$

where, for every set of atomic and structured attributes $\bar{B}$ and $\bar{Y}$, and for every nested database D containing no two relations with identical scheme,

$$[\bar{B}\bar{Y}, D]^{iso} = \nu_{\Delta(\bar{B}\bar{Y})}([\bar{B}, D], \{\mu_Y([\bar{B}Y, D]) \mid Y \in \bar{Y}\}),$$
$$[\bar{B}\bar{Y}, D] = \pi_{\bar{B}\bar{Y}}(\bowtie_{B\in\bar{B}}[B, D]_1 \bowtie \bowtie_{Y\in\bar{Y}}[Y, D]_1),$$

and where, for every scheme Δ with root $\bar{A}\bar{Y}X_1 \ldots X_n$ and for every relations $\mathcal{R}, \mathcal{R}_1, \ldots, \mathcal{R}_n$ with schemes $\Delta(\bar{A}\bar{Y}), \Delta_{X_1}, \ldots, \Delta_{X_n}$ respectively, if X_i are defined in Δ by $X_i \equiv \bar{Q}_i$ $(1 \le i \le n)$,

$$\nu_\Delta(\mathcal{R}, \{\mathcal{R}_1, \ldots, \mathcal{R}_n\}) = \{t \mid \text{for some } u \in \mathcal{R}, \text{ for every } i \in \{1, \ldots, n\}$$
$$t[\bar{A}\bar{Y}] = u,$$
$$t[X_i] = \{v[\bar{Q}_i] \mid v \in \mathcal{R}_i \text{ and } v[\bar{A}] = u[\bar{A}]\}\}.$$

Figure 2: Restructuring operator – Version II

tribute of $sub(X', \Delta')$ (resp. $sub(X, \Delta)$). That condition is formalized in the following definitions.

Definitions. Let X_1 and X_2 be two structured attributes defined in relation schemes Δ_1 and Δ_2 respectively. Let $\bar{A}_1$, $\bar{A}_2$, $\bar{B}_1$, and $\bar{B}_2$ be the sets of atomic attributes appearing in Δ_1, Δ_2, $sub(X_1, \Delta_1)$, and $sub(X_2, \Delta_2)$ respectively. Then, X_1 in Δ_1 is *restructurable* to X_2 in Δ_2 if X_1 is not isomorphic to X_2 and if $\bar{B}_1 \cap \bar{A}_2 = \bar{B}_2 \cap \bar{A}_1$. X_1 in Δ_1 is *maximally restructurable* to X_2 in Δ_2 if there are no attribute Y_1 in Δ_1 restructurable to Y_2 in Δ_2 such that X_1 and X_2 are defined in $sub(Y_1, \Delta_1)$ and $sub(Y_2, \Delta_2)$ respectively.

The set of structured attributes in Δ_1 that are maximally restructurable to attributes in Δ_2 is denoted $struct(\Delta_1, \Delta_2)$. ◇

Version III of the restructuring operator is similar to Version II except that sub-relations of $\mathcal{R}$ with schemes $sub(X, \Delta)$ such that X in Δ is maximally restructurable to an attribute X' in Δ' are restructured to $sub(X', \Delta')$ as soon as they appear during the flattening process. Combined with the absence of flattening for isomorphic attributes, the flattening process then results in a nested relational database $\Phi^{struct}_\Delta(\mathcal{R}, \Delta')$ where sub-relations of R with schemes isomorphic to sub-schemes of Δ' have been renamed and where sub-relations of R with schemes restructurable to sub-schemes of Δ' have been restructured to these sub-schemes.

The new restructuring operator is given in Figure 3 (definitions of $[\_,\_]^{iso}$ and ν_Δ are those of Version II in Figure 2).

Let $\mathcal{R}$ be a PNF relation in $\boldsymbol{P}_\Delta$ with root $\bar{A}\bar{X}$,
Δ' be a relation scheme with root $\bar{A}'\bar{X}'$,
$\bar{X}_1 = iso(\Delta, \Delta')$ and $\bar{X}'_1 = iso(\Delta', \Delta)$,
$\bar{X}_2 = struct(\Delta, \Delta')$ and $\bar{X}'_2 = struct(\Delta', \Delta)$,
$\bar{X}_3 = \bar{X} \setminus (\bar{X}_1 \cup \bar{X}_2)$ and $\bar{X}'_3 = \bar{X}' \setminus (\bar{X}'_1 \cup \bar{X}'_2)$,

$$R^{struct}_{\Delta'}(\mathcal{R}) = R^{s}_{\Delta\to\Delta'}(\Phi^{struct}_\Delta(\mathcal{R}, \Delta')),$$

$$R^{s}_{\Delta\to\Delta'}(D) = \nu_{\Delta'}([\bar{A}'\bar{X}'_1\bar{X}'_2, D]^{iso}, \{R^{s}_{\Delta\to\Delta'_{X'_3}}(D) \mid X'_3 \in \bar{X}'_3\}),$$

$$\Phi^{struct}_\Delta(\mathcal{R}, \Delta') = \{R^{loc}_{\Delta'}(\mathcal{R})\} \cup \bigcup_{X\in\bar{X}_3} \Phi^{struct}_{\Delta_X}(\mu_X\pi_{\bar{A}X}(\mathcal{R}), \Delta'),$$

where, if, for every X_1 and X_2 belonging to $\bar{X}_1$ and $\bar{X}_2$ respectively, X'_1 denotes the attribute in Δ' that is maximally isomorphic to X_1 in Δ, X'_2 denotes the attribute in Δ' that is maximally restructurable to X_2 in Δ,

$$R^{loc}_{\Delta'}(\mathcal{R}) = \{t \mid \text{for some } u \in \mathcal{R}, \text{ for every } X_1 \in \bar{X}_1, X_2 \in \bar{X}_2,$$
$$t[\bar{A}] = u[\bar{A}],$$
$$t[X'_1] = \rho_{sub(X_1,\Delta)\to sub(X'_1,\Delta')}(u[X_1]),$$
$$t[X'_2] = R^{struct}_{sub(X'_2,\Delta')}(u[X_2]) \quad \}.$$

Figure 3: Restructuring operator – Version III

The following theorem proves the equivalence between Version III and Version I of the restructuring operator.

Theorem 4. *For every relation scheme Δ' and every PNF relation $\mathcal{R}$,*

$$R^{struct}_{\Delta'}(\mathcal{R}) = R_{\Delta'}(\mathcal{R}).$$

4 Comparisons and conclusions

The problem of data restructuring was studied by Hull and Yapp ([7]) for a very large class of hierarchical data structures. It was also considered by Abiteboul and Bidoit in a more restricted data model, the Verso model ([2], [1]) for which an algebra incorporating restructuring was described.

The data structures of the Verso model (called *Verso instances*) are PNF relations. Contrary to the common practice in nested models, restructuring in the Verso model is not performed by nestings and unnestings but by a general restructuring operator.

Let $\mathcal{R}$ be a Verso instance with scheme Δ. The *set of facts associated with* $\mathcal{R}$ is $fact(\mathcal{R}) = \Pi(*(\Phi_\Delta(\mathcal{R})))$ where, if D is a flat relational database,

- $\Pi(D)$ is the *closure of* D *under projection*, i.e.,

$$\Pi(D) = \{\pi_{\bar{A}}(S) \mid S \in D \text{ and } \bar{A} \subset scheme(S)\}.$$

- $*(D)$ is the *closure of* D *under join*, i.e.,

$$*(D) = \bigcup_{i=0}^{\infty} D_i,$$

where $D_0 = D$ and $D_{i+1} = \{S \bowtie S_i \mid S \in D \text{ and } S_i \in D_i\}$.

The restructuring of $\mathcal{R}$ to scheme Δ' is then defined, in the Verso model, as the greatest instance $\mathcal{R}'$ (if it exists) with scheme Δ' such that $fact(\mathcal{R}') \subset fact(\mathcal{R})$.

Restructuring is thus not always defined. For example, it is easy to verify that the Verso instance $\mathcal{R} = \{< a, b, c >, < a, b', c' >\}$ with scheme $\{A, B, C\}$ cannot be restructured to scheme $\{\{A, X, Y\}, X \equiv B, Y \equiv C\}$. Indeed, $\mathcal{R}_1 = \{< a, \{b\}, \{c\} >\}$ and $\mathcal{R}_2 = \{< a, \{b'\}, \{c'\} >\}$ are such that $fact(\mathcal{R}_1) \subset fact(\mathcal{R})$ and $fact(\mathcal{R}_2) \subset fact(\mathcal{R})$ but $\mathcal{R}_3 = \mathcal{R}_1 \sqcup \mathcal{R}_2 = \{< a, \{b, b'\}, \{c, c'\} >\}$ is such that $fact(\mathcal{R}_3) \supset fact(\mathcal{R})$.

We think that there are no reasons for banning such restructurings. With our algorithm, the resulting relation would be $\{< a, \{b, b'\}, \{c, c'\} >\}$. If tuple $< a, b, c >$ represents the information "department a receives part b from supplier c", the tuple $\{< a, \{b, b'\}, \{c, c'\} >\}$ is meaningful and can be translated to: "a is a department, the set of parts it receives is $\{b, b'\}$ and the set of its suppliers for those parts is $\{c, c'\}$".

We believe (but we have not looked for a formal proof) that, when it is defined, restructuring in the Verso model is equivalent to restructuring with our algorithm. In conclusion, our approach to restructuring is more general than the approach followed in the Verso model.

Up to our knowledge, Abiteboul and Bidoit have never described any algorithm for computing restructurings. They have only given the definition recalled above and have been mainly interested in finding which restructurings were lossless operations. The conditions they have found for restructurings to be without loss of information are not either operational.

Our main contributions have thus been to give a natural and broader definition to the concept of data restructuring for PNF relations, valid for any pair of relation schemes, and to present an operational description of how data restructuring can be effectively (and efficiently) performed.

Restructurings without loss of information is worth a closer look. First of all, a clear definition of the concept of lossless restructuring is needed. That concept is clear when the origin and the target schemes share the same set of atomic attributes. The situation is more complicated when atomic attributes appear or disappear during restructuring. Then, the main issue can be tackled: under which conditions is restructuring performed without loss of information? It clearly appeared in previous examples that the absence of empty values for structured attributes decreased the information loss during restructuring. The presence of functional and join dependencies can also have the same effect. It would be interesting to study the impact of such constraints on restructurings.

We are currently investigating a theoretical justification for our definition of $R_{\Delta'}$. Let $\mathcal{R}$ be a relation in $\boldsymbol{P}_\Delta$ to be restructured to Δ'. First, an order relation $\prec_{\mathcal{R}}$ must be defined in $\boldsymbol{P}_{\Delta'}$ such that, for every $\mathcal{R}_1$ and $\mathcal{R}_2$ in $\boldsymbol{P}_{\Delta'}$, $\mathcal{R}_1 \prec \mathcal{R}_2$ formalizes the idea that $\mathcal{R}_2$ is a better restructuring of $\mathcal{R}$ than $\mathcal{R}_1$. Next, $R_{\Delta'}(\mathcal{R})$ must be proved to be maximal in $\boldsymbol{P}_{\Delta'}$ for $\prec_{\mathcal{R}}$.

ACKNOWLEDGEMENTS

This work was partly supported by the Commission of the European Communities, under Project ESTEAM-316 of the ESPRIT Program. We are grateful to P. Gribomont, A. Pirotte and D. Roelants, our colleagues of PRLB, for their careful readings of early drafts of this paper and useful comments.

References

[1] S. Abiteboul and N. Bidoit. Non first normal form relations: An algebra allowing data restructuring. *Journal of Computer and System Sciences*, 33:361–393, 1986.

[2] S. Abiteboul and N. Bidoit. Non first normal form relations to represent hierchically organized data. In *Proc. 3rd ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 191–200, Waterloo, Ontario, Canada, April 1984.

[3] F. Bancilhon, P. Richard, and M. Scholl. On line processing of compacted relations. In *Proc. 8th International Conference on Very Large Databases*, pages 263–269, Mexico City, Sep. 1982.

[4] P. Dadam, K. Kuespert, F. Anderson, H. Blanken, R. Erbe, J. Guenauer, V. Lum, P. Pistor, and G. Walsh. A DBMS

prototype to support extended NF^2 relations: An integrated view on flat tables and hierarchies. In *Proc. ACM-SIGMOD International Conference on Management of Data*, pages 356–367, Washington D.C., May 1986.

[5] G. Hulin. *On Restructuring Nested Relations in Partitioned Normal Form.* Technical Report, Philips Research Laboratory Brussels, 1990.

[6] G. Hulin. *A Relational Algebra for Nested Relations in Partitioned Normal Form.* Manuscript M318, Philips Research Laboratory Brussels, Oct. 1989.

[7] R. Hull and C. Yap. The format model: A theory of database organization. *Journal of the ACM*, 31(3):518–537, July 1984.

[8] G. Jaeschke and H. Schek. Remarks on the algebra of non first normal form relations. In *Proceedings of ACM Symposium on Principles of Database Systems*, pages 124–137, Los Angeles, March 1982.

[9] D. Maier. *The Theory of Relational Databases.* Pitman Publ., London, 1983.

[10] A. Makinouchi. A consideration on normal form of not-necessarily-normalized relation in the relational data model. In *Proc. 3rd International Conference on Very Large Databases*, pages 447–453, Tokyo, Oct. 1977.

[11] J. Paredaens, P. De Bra, M. Gyssens, and D. Van Gucht. *The Structure of the Relational Database Model*, chapter 7. Volume 17 of *EATC Monographs on Theoretical Computer Science*, Springer-Verlag, 1989.

[12] P. Pistor and F. Andersen. Designing a generalized NF^2 model with an SQL-type language interface. In *Proc. 12th International Conference on Very Large Databases*, pages 278–285, Kyoto, Aug. 1986.

[13] P. Pistor and R. Traunmueller. A database language for sets, lists and tables. *Information Systems*, 11(4):323–336, 1986.

[14] M. A. Roth, H. F. Korth, and D. S. Batory. SQL/NF: A query language for ¬1NF relational databases. *Information Systems*, 12(1):99–114, 1987.

[15] M. A. Roth, H. F. Korth, and A. Silberschatz. Extended algebra and calculus for nested relational databases. *ACM Transactions on Database Systems*, 13(4):389–417, Dec. 1988.

[16] H. Schek and P. Pistor. Data structures for an integrated data base management and information retrieval system. In *Proc. 8th International Conference on Very Large Databases*, pages 197–207, Mexico City, Sep. 1982.

[17] H. Schek and M. Scholl. The relational model with relation-valued attributes. *Information Systems*, 11(2):137–147, 1986.