A Uniform Approach toward Handling Atomic and Structured Information in the Nested Relational Database Model

MARC GYSSENS AND JAN PAREDAENS

University of Antwerp, Antwerp, Belgium

AND

DIRK VAN GUCHT

Indiana University, Bloomington, Indiana

Abstract. The algebras and query languages for nested relations defined thus far do not allow us to "flatten" a relation scheme by disregarding the internal representation of data. In real life, however, the degree in which the structure of certain information, such as addresses, phone numbers, etc., is taken into account depends on the particular application and may even vary in time. Therefore, an algebra is proposed that does allow us to simplify relations by disregarding the internal structure of a certain class of information. This algebra is based on a careful manipulation of attribute names. Furthermore, the key operator in this algebra, called "copying," allows us to deal with various other common queries in a very uniform manner, provided these queries are interpreted as operations on classes of semantically equivalent relations rather than individual relations. Finally, it is shown that the proposed algebra is complete in the sense of Bancilhon and Paredaens.

Categories and Subject Descriptors: H.2.1 [**Database Management**]: Logical Design—*data models*; H.2.3 [**Database Management**]: Languages—*query languages*

General Terms: Design, Languages, Theory, Verification

Additional Key Words and Phrases: Algebra, atomic and nonatomic values, copying, expressiveness, nested and flat relations and databases, prime values, relation classes

1. *Introduction*

When Codd introduced the relational model in 1970–71 [5, 6], he also defined a number of normal forms to avoid redundancy and update anomalies. The first normal form required a relation to consist of only "atomic" (i.e., nonstructured) values, whereas the other ones involved the presence of certain constraints. Although these other normal forms have been the subject of a broad discussion about their desirability, it had been taken for granted for a long time that a relation must always be in first normal form.

The work of M. Gyssens was supported in part by the Belgian National Fund for Scientific Research.

Authors' present addresses: M. Gyssens and J. Paredaens, Department of Mathematics and Computer Science, University of Antwerp (UIA), Universiteitsplein 1, B-2610 Antwerp, Belgium; D. Van Gucht, Computer Science Department, Indiana University, Lindley Hall 101, Bloomington, IN 47405-4101.

During the last years however, much attention has been paid to nonfirst normal form relations, also called nested relations [12, 16, 24, 26]. It is nowadays indeed agreed upon that the traditional "flat" relational model of Codd, although very tractable and hence attractive from a theoretical viewpoint, lacks the capacity to structure data sufficiently as to match reality. Therefore many models have been designed to deal with structured data. For these models, query languages in the form of algebras [1, 2, 9, 12, 14, 17, 25, 26], calculi [1, 14,17, 20, 24], and extended SQL [21, 22] have been proposed. They usually consist of the obvious generalizations of the classical relational algebra, calculus, and SQL, augmented with constructs to explicitly restructure data.

Recently, generalizations of results concerning the expressiveness of flat relational query languages [3, 4, 7, 18] have been obtained [1, 10, 11, 17, 20, 24, 28]. For example, in [28], it is shown that a particular algebra which manipulates nested relations is complete in the sense Bancilhon and Paredaens [3, 18]. This notion, called BP-completeness [4], is concerned with the decision problem whether for a given query language some relation instance (be it flat or nested) can be obtained from another using an expression in that query language.

As mentioned before, the original motivation for considering nested relations was abandoning Codd's condition of "first normal form." It should however be readily seen that first normal form is not a purely mathematical condition upon a relation, but rather a philosophical one. Data that are considered as structured by some people might be atomic for others. Good examples of such data types are addresses and phone numbers. Therefore, it is our opinion that an algebra for nested relations must necessarily have an operator that allows one to disregard the structure of certain data and consider them as atomic. We propose a theoretical model for such an algebra and extend the completeness result of [28]. Along the way, we also include in our model the possibility for structured attributes to have the empty set as a value (cf., also [2, 22, 23, 25]). This possibility allows us to model various real-life situations (such as a parent having no children) without being forced to introduce null values [8, 29]. We feel that by adding these features, for example, to the algebra of [26], a more unified formalism for manipulating nested relations is obtained. Furthermore, we show that each nested relation can be encoded in a natural way by a flat relation, within the framework of the algebra we propose. Finally, we use this last result to show completeness of our algebra in the sense of Bancilhon and Paredaens.

This article is organized as follows. In Section 2, we review some terminology and results (in particular, concerning BP-completeness) for the flat relational algebra, needed in the sequel. In Section 3, we propose a formal model to represent nested relations and in Section 4, we define our algebra. From the discussions in Section 4, it follows that algebraic operations should be defined on classes of relations, rather than individual relations. This idea is developed in Section 5. Finally, in Section 6, we examine the expressiveness of our algebra with respect to BP-completeness.

2. *The Flat Relational Model*

The flat relational model was introduced by Codd [5, 6]. In this section, we only review the notations and definitions that are used in the sequel; a global discussion of the relational model can be found in [15], [19], [26], and [27]. Essentially, a flat relational database consists of a collection of flat relations, each of which can be represented as a flat table.

Example 2.1. Table I represents a relation from a social security database indicating parents (with their first and last name) together with their age, their jobs, their children and the age of their children.

Notice that this relation is neither in second nor in third normal form [6]. Furthermore, persons with a job, but without children cannot be represented in this relation without using null values [8, 29].

Definition 2.1. Consider an infinitely enumerable set U, called the *set of all attributes*, and an infinitely enumerable set of values V, called the *set of all values*. A *flat relation scheme* Ω is a finite subset of U. A *tuple over* Ω is a mapping from Ω into V; $\mathscr{T}_\Omega = V^\Omega$ denotes[1] the set of all tuples over Ω. An *instance over* Ω is a finite set of tuples over Ω; $\mathscr{I}_\Omega = 2^{(\mathscr{T}_\Omega)}$ denotes[2] the set of all instances over Ω. A *flat relation* is a pair (Ω, ω) where $\Omega \subseteq U$ is a flat relation scheme and $\omega \in \mathscr{I}_\Omega$ an instance over Ω. A *flat database* is a finite set of flat relations.

A number of well-known unary and binary operations are defined on flat relations. They form the so-called flat relational algebra.

Definition 2.2. The *flat relational algebra* consists of the union, the difference, the Cartesian product, the projection, the renaming, and the selection.

—Given two flat relations (Ω, ω_1) and (Ω, ω_2), defined over the same scheme Ω. The *union* and *difference* are defined by

$$(\Omega, \omega_1) \cup (\Omega, \omega_2) = (\Omega, \omega_1 \cup \omega_2),$$
$$(\Omega, \omega_1) - (\Omega, \omega_2) = (\Omega, \omega_1 - \omega_2).$$

—Given two flat relations (Ω_1, ω_1) and (Ω, ω_2) with $\Omega_1 \cap \Omega_2 = \varnothing$. The *Cartesian product* $(\Omega_1, \omega_1) \times (\Omega_2, \omega_2)$ is defined as $(\Omega_1 \cup \Omega_2, \omega')$ where

$$\omega' = \{t \in \mathscr{T}_{\Omega_1 \cup \Omega_2} \mid t|_{\Omega_1} \in \omega_1 \ \& \ t|_{\Omega_2} \in \omega_2\}.$$

—Given a flat relation (Ω, ω) and $\Omega' \subseteq \Omega$. The *projection* $\pi_{\Omega'}(\Omega, \omega)$ equals (Ω', ω') where

$$\omega' = \{t|_{\Omega'} \mid t \in \omega\}.$$

—Given a flat relation (Ω, ω), $B \in \Omega$ and $A \in U - \Omega$. The *renaming* $\rho_{A \leftarrow B}(\Omega, \omega)$ equals $(\Omega - \{B\} \cup \{A\}, \omega')$ where

$$\omega' = \{t' \in \mathscr{T}_{\Omega - \{B\} \cup \{A\}} \mid \exists t \in \omega : t'|_{\Omega - \{B\}} = t|_{\Omega - \{B\}} \ \& \ t'(A) = t(B)\}.$$

—Given a flat relation (Ω, ω) and $A, B \in \Omega$. The *selection* $\sigma_{A=B}(\Omega, \omega)$ equals (Ω, ω') where

$$\omega' = \{t \in \omega \mid t(A) = t(B)\}.$$

Expressions of the flat algebra are then defined in the usual manner. To avoid extensive use of brackets, we adopt the following convention about the *precedence among the operators*

(1) unary operators
(2) Cartesian product
(3) set-operators

[1] For arbitrary sets S and T, we denote by S^T the set of all total mappings from T into S.
[2] For an arbitrary set S, we denote by $2^{(S)}$ the set of all finite subsets of S.

TABLE I

P-FNAME	P-LNAME	P-AGE	JOB	CHILD	C-AGE
Jeff	Willows	54	professor	Glenda	23
Jeff	Willows	54	professor	Mark	21
Jeff	Willows	54	professor	Rita	21
Jeff	Willows	54	manager	Glenda	23
Jeff	Willows	54	manager	Mark	21
Jeff	Willows	54	manager	Rita	21
Mary	Higgins	49	secretary	Glenda	23
Mary	Higgins	49	secretary	Mark	21
Mary	Higgins	49	secretary	Rita	21

Example 2.2. Let us denote Ω = {P-FNAME, P-LNAME, P-AGE, JOB, CHILD, C-AGE} and let ω be the instance represented by Table I in Example 1.1. We could ask for the relation of all pairs of children with the same age. Let $\{C1, C2\}$ be the scheme of this relation. Then the following flat relational algebra expression can be used to compute it:

$$\rho_{C1 \leftarrow \text{CHILD}}\pi_{\{\text{CHILD},C2\}}\sigma_{\text{C-AGE}=A2}$$
$$(\pi_{\{\text{CHILD,C-AGE}\}}(\Omega, \omega) \times \rho_{C2 \leftarrow \text{CHILD}}\rho_{A2 \leftarrow \text{C-AGE}}\pi_{\{\text{CHILD,C-AGE}\}}(\Omega, \omega)).$$

Note that we did not include intersection, since it can be defined in terms of the difference. For the same reason, we did not mention selection on inequality. Neither did we consider other types of selection, since they require additional information (e.g., an order relation on V). In the same spirit, we do not introduce constants; if they are really needed, they can be represented by relations, each containing precisely one constant.

In [18], a notion called BP-completeness [4], was introduced to measure the expressiveness of the flat relational algebra. We recall here the main definitions and results of that paper, adapted to our formalism. (A similar notion has been introduced in [3] for the relational calculus.)

Definition 2.3. Let $r = (\Omega, \omega)$ be a flat relation. The *set of values of* r is defined as

$$\text{val}(r) = \{t(A) \mid t \in \omega \ \& \ A \in \Omega\}.$$

Let $d = \{r_1, \ldots, r_n\}$ be a flat database. The *set of values of* d is defined as

$$\text{val}(d) = \bigcup_{i=1}^{n} \text{val}(r_i).$$

The *basic information of* r, denoted BI(r), is the set consisting of all flat relations s for which there exists an expression $E(x)$ in the flat relational algebra such that $E(r) = s$. The *basic information of* d, denoted BI(d), is the set consisting of all flat relations s for which there exists an expression $E(x_1, \ldots, x_n)$ in the flat relational algebra such that $E(r_1, \ldots, r_n) = s$.

The notion of BP-completeness as a measure for expressiveness was concerned with the decision problem whether some given flat relation can be expressed by the flat relational algebra from another given flat relation. In order to define BP-completeness in a satisfactory way, we need the notion of cogroup.

Definition 2.4. Let $r = (\Omega, \omega)$ be a flat relation. The *cogroup of r*, denoted CG(r), is defined by

$$\mathrm{CG}(r) = \{\psi \mid \psi \text{ is a permutation of val}(r) \text{ and } \psi(r) = r\}.$$

Let $d = \{r_1, \ldots, r_n\}$ be a flat database. The *cogroup of d*, denoted CG(d), is defined by

$$\mathrm{CG}(d) = \{\psi \mid \psi \text{ is a permutation of val}(d) \text{ and } \forall i = 1, \ldots, n : \psi(r_i) = r_i\}.$$

(ψ is extended to tuples, instances, and relations in the natural way.)

Example 2.3. The set of values of the relation represented in Example 2.1 consists of 14 elements: *Jeff*, *Mary*, *Willows*, *Higgins*, *54*, *49*, *professor*, *manager*, *secretary*, *Glenda*, *Mark*, *Rita*, *23*, and *21*.

It can be easily seen that the permutation interchanging the values *professor* and *manager* and leaving the other values fixed, belongs to the cogroup of the relation in Example 2.1. We invite the reader to verify that the entire cogroup of that relation consists of four permutations.

It is our aim to decide membership of the basic information of a flat relation of some other flat relation by comparing their cogroups. In [18] it has been shown that

THEOREM 2.1. *Let r and s be flat relations.* $s \in BI(r)$ *if and only if* $val(s) \subseteq v(r)$ *and* $CG(r)|_{val(s)} \subseteq CG(s)$, *where* $CG(r)|_{val(s)}$ *stands for* $\{\psi|_{val(s)} \mid \psi \in CG(r)\}$.

It goes without saying that the notion of basic information can be generalized to query languages other than the flat relational algebra. If Theorem 2.1 holds for such a query language, we say that it is *BP-complete*. In particular, the flat relational algebra is BP-complete.

3. *The Nested Relational Model*

Unfortunately, the flat relational model is often inadequate for a natural representation of data, as was observed by several researchers (e.g., [12, 16, 26]). They suggested to drop the first normal form condition of Codd [6]. Indeed, if we have a look at, for example, the relation represented in Example 2.1, it should be readily seen that this relation is not the most ideal way to represent the data under consideration; for example, it is not in second normal form. Rather than to go for a decomposition of this relation, we should group together data that are closely related.

Example 3.1. A relation relating parents with their first and last name, age, and professions to their children with their respective names and ages could also be represented as shown in Table II.

A relation, such as the one represented in Table II, is called a *nested relation*. At the highest level, the nested relation considered above should be interpreted as a two-tuple relation over a two-attribute scheme. Note that the data of the relation in Example 2.1 are all represented in the first tuple of this relation. Furthermore, the data in the second tuple of this relation could not possibly be represented in the relation of Example 2.1, unless null values are permitted.

As is explained in the introduction, it is the main purpose of this section to present a theoretical model for nested relations that is suited to extend the nested

TABLE II

{{P-FNAME	P-LNAME}	P-AGE	{JOB }	{CHILD	C-AGE}}
Jeff	Willows	54	professor manager	Glenda	23
				Mark	21
				Rita	21
Mary	Higgins	49	secretary		
Tom	Jenkins	32	shopkeeper		
Brenda	Jones	31			

relational algebra (e.g., [28]) in such a way that

—it is possible to disregard the internal representation of nested data;
—higher level attributes can be emptyset-valued.

Basically, as in the flat relational model, we assume that there is an infinitely enumerable set U of *atomic attributes* and an infinitely enumerable set V of *atomic values*. From these atomic attributes and values, we construct nested attributes and values as well as nested relation schemes, instances, and relations. First, we explain how nested attributes are constructed.

Nested attributes can either be *atomic* or *nonatomic*. Atomic attributes are simply the elements of U. Nonatomic attributes are finite sets of nested attributes, which in turn can be either atomic or nonatomic. However, in order to avoid ambiguity, we impose one restriction: No atomic attribute may appear twice in a nested attribute. For example, if $A, B, C \in U$, then $\{A, \{B, C\}\}$ is a nested attribute, but $\{A, \{A, B\}\}$ is not.

Example 3.2. Reconsider the nested relation represented in Example 3.1. We already mentioned that this relation has two attributes at the highest level. These nested attributes are {{P-FNAME, P-LNAME}, P-AGE, {JOB}}, and {CHILD, C-AGE}. Both attributes are nonatomic; the former consists of the nonatomic attribute {P-FNAME, P-LNAME}, the atomic attribute P-AGE, and the nonatomic attribute {JOB}, whereas the latter only consists of atomic attributes.

We now formally define the set $\mathscr{U}$ of nested attributes. At the same time, we introduce the function ato: $\mathscr{U} \to 2^{(U)}$ returning for each element of $\mathscr{U}$ the set of all atomic attributes that appear in it.

Definition 3.1. The *set of nested attributes* $\mathscr{U}$ and the function ato: $\mathscr{U} \to 2^{(U)}$ are recursively defined as follows:

(1) $\emptyset \in \mathscr{U}$ and $\text{ato}(\emptyset) = \emptyset$;
(2) $U \subseteq \mathscr{U}$ and $\forall A \in U: \text{ato}(A) = \{A\}$;
(3) If $X_1, \ldots, X_n \in \mathscr{U}$ and $\forall i, j = 1, \ldots, n$ with $i \neq j$, $\text{ato}(X_i) \cap \text{ato}(X_j) = \emptyset$, then $\{X_1, \ldots, X_n\} \in \mathscr{U}$. Furthermore, $\text{ato}(\{X_1, \ldots, X_n\}) = \bigcup_{i=1}^{n} \text{ato}(X_i)$;
(4) No other elements are in $\mathscr{U}$.

A nested attribute of U is called an *atomic attribute*; a nested attribute of $\mathcal{U} - U$ is called a *nonatomic attribute.*

Note that according to Definition 3.1, the empty set can be considered as a nonatomic attribute. Although admittedly it has little practical relevancy to consider relations in which the empty set appears as an attribute, we do not want to exclude this possibility formally. The reason is twofold: first, because we want to build a model with as few restrictions as possible and second, because we use this possibility to construct empty relations over arbitrary schemes, as will be seen later.

The definition of a nested relation scheme is now very straightforward.

Definition 3.2. A *nested relation scheme* Ω is a nonatomic nested attribute, that is, an element of $\mathcal{U} - U$. In particular, if $\Omega \subseteq U$, then U is called a *flat scheme.*

Note that any finite set of atomic attributes is a nonatomic attribute; hence, Definition 3.2 is indeed a generalization of Definition 2.1.

Example 3.3. The scheme of the nested relation in Example 3.1 is the nonatomic attribute

{{{P-FNAME, P-LNAME}, P-AGE, {JOB}}, {CHILD, C-AGE}}.

Notice that elements of $\mathcal{U} - U$ can be alternatively seen as

(1) nonatomic nested attributes;
(2) nested relation schemes.

We often use this dualism in the sequel.

Of course, we would like to define nested relation instances in such a way that flat relation instances are a special case of it. Hence we define a nested relation instance over a relation scheme Ω as a finite set of tuples over Ω. If t is such a tuple, and $X \in \Omega$ is a nonatomic attribute, then we require $t(X)$ to be a nested relation instance over X, the latter being interpreted as a nested relation scheme.

Example 3.4. Reconsider the relation of Example 3.1. The first component in each of both tuples must be regarded as a nested relation instance over the scheme {{P-FNAME, P-LNAME}, P-AGE, {JOB}}. The first and the last component of each tuple in these instances are in turn (flat) relation instances over the schemes {P-FNAME, P-LNAME} and {JOB}, respectively. Similarly, the second component in each of both tuples of the nested relation instance of Example 3.1 is a (flat) relation instance over the scheme {CHILD, C-AGE}.

Formally, a tuple is a mapping from a set of attributes to a set of values. From our discussion above, the *set of nested values* $\mathcal{V}$ must contain, apart from the atomic values, all nested relation instances. We now define

Definition 3.3. The set $\mathcal{V}$ of *nested values*, the set $\mathcal{I}_X$ of all *nested relation instances over* $X \in \mathcal{U} - U$, the set $\mathcal{T}_X$ of all *tuples over* $X \in \mathcal{U} - U$ and the set $\mathcal{I}$ of all *nested relation instances* are the smallest sets satisfying:

(1) $\mathcal{V} = V \cup \mathcal{I}$ (V being the set of all atomic values);
(2) $\mathcal{I} = \bigcup_{X \in \mathcal{U} - U} \mathcal{I}_X$;
(3) $\mathcal{I}_X$ consists of all finite subsets of $\mathcal{T}_X$;
(4) $\mathcal{T}_X$ consists of all mappings t from X into $\mathcal{V}$ satisfying $t(Y) \in \mathcal{I}_Y$ for all nonatomic attributes $Y \in X - U$.

Note that the set of nested values $\mathcal{V}$ is closed in the sense that if the set V in Definition 3.3 is replaced by $\mathcal{V}$, no larger set will be found (as should be the case).

Consequently Definition 3.3 suggests the introduction of the following operator on sets of values, that will be used often in the sequel:

Definition 3.4. Let $W \subseteq \mathscr{V}$ be a set of nested values. If we apply Definition 3.3 with V replaced by W, we obtain a set $\mathscr{W}$. $\mathscr{W}$ is said to be the *set of nested values generated by* W and denoted as $\text{gen}(W)$.

From Definitions 3.3 and 3.4, we immediately deduce that

—If $W \subseteq W' \subseteq \mathscr{V}$, then $\text{gen}(W) \subseteq \text{gen}(W')$;
—For all $W \subseteq \mathscr{V}$, $\text{gen}(\text{gen}(W)) = \text{gen}(W)$;
—$\text{gen}(V) = \mathscr{V} = \text{gen}(\mathscr{V})$.

We now have all the necessary ingredients to formally define a nested relation.

Definition 3.5. A *nested relation*, or briefly a relation, is a pair (Ω, ω) where $\Omega \in \mathscr{U} - U$ is a nested relation scheme and $\omega \in \mathscr{I}_\Omega$ a nested relation instance over Ω. (Ω, ω) is called *flat* if Ω is a flat scheme, that is, if $\Omega \subseteq U$. A *nested database*, or briefly a database, is a finite set of nested relations. It is called *flat* if all of its relations are flat.

Clearly, Definition 3.5 can be considered as a generalization of Definition 2.2.

Note that, according to Definition 3.3, values corresponding to atomic attributes need *not* be atomic; they can also be relation instances. In the latter case, however, we assume that the internal structure of that instance is inaccessible. This is a reasonable assumption, since, whenever we want to access the internal representation of a value in a database, we have to do that through the corresponding attribute, which serves as an identifier for that value in the tuple under consideration.

Why do we allow these inaccessible relations as values corresponding to atomic attributes? As explained earlier, we wish to introduce an operator in the following section that allows us to disregard the internal representation of a nonatomic value (which is a relation instance). This can be accomplished by "renaming" the nonatomic attribute to which this instance corresponds into an atomic one. Indeed, as explained above, the internal structure of the instance will then have become inaccessible. We illustrate this idea with an example.

Example 3.5. Reconsider the nested relation in Example 3.1. One might wish to disregard the fact that a person's name consists of a first and a last name and rather regard the entire name of a person as an unstructured datum. To achieve this goal, we "rename" the nonatomic attribute {P-FNAME, P-LNAME} in the scheme of the relation in Example 3.1 to the atomic attribute PARENT. The resulting relation could be represented as shown in Table III.

As explained earlier, the first component of each of both tuples in the above nested relation instance, is a nested relation instance over the scheme {PARENT, P-AGE, {JOB}}. In each tuple of each of both instances, the first component is in turn a (flat) relation instance over the scheme {P-FNAME, P-LNAME},[3] although the corresponding attribute PARENT is atomic.

In summary, the values in a given relation of which the internal structure is either disregarded or does not exist, are precisely those values that correspond to atomic attributes. These values will be called quite fittingly the *prime values* of that

[3] Since this scheme is not any longer subsumed by the nested relation scheme of the global instance, we had to mention it explicitly in the above table, each time an instance over that scheme occurred.

TABLE III

{{PARENT		P-AGE	{JOB }}	{CHILD	C-AGE}}
{P-FNAME	P-LNAME}	54	professor manager	Glenda	23
Jeff	Willows			Mark	21
				Rita	21
{P-FNAME	P-LNAME}	49	secretary		
Mary	Higgins				
{P-FNAME	P-LNAME}	32	shopkeeper		
Tom	Jenkins				
{P-FNAME	P-LNAME}	31			
Brenda	Jones				

relation. Note in particular that the designation "prime value" makes only sense in the context of a relation. Formally, we define

Definition 3.6. Let $r = (\Omega, \omega)$ be a nested relation. The *set val(r) of all prime values of r* is recursively defined by

$$\mathrm{val}(r) = \bigcup_{t \in \omega} \left(\left(\bigcup_{A \in \Omega \cap U} t(A) \right) \cup \left(\bigcup_{X \in \Omega - U} \mathrm{val}(t(X)) \right) \right).$$

Let $d = \{r_1, \ldots, r_n\}$ be a nested database. The *set val(d) of all prime values of d* is defined as

$$\mathrm{val}(d) = \bigcup_{i=1}^{n} \mathrm{val}(r_i).$$

Observe that, when Definition 3.6 is applied to the flat relational model of Section 2, we obtain the function val(r) (respectively, val(d)) of Definition 2.3. This justifies the use of the same name for both functions.

Example 3.6. In the relation of Example 3.1, the prime values are the atomic values *Jeff*, *Mary*, *Tom*, *Brenda*, *Willows*, *Higgins*, *Jenkins*, *Jones*, *54*, *49*, *32*, *31*, *professor*, *manager*, *secretary*, *shopkeeper*, *Glenda*, *Mark*, *Rita*, *23*, and *21*. There are no other prime values in that relation, since no other value corresponds to an atomic attribute.

In the relation of Example 3.5, the prime values are the atomic values *54*, *49*, *32*, *31*, *professor*, *manager*, *secretary*, *shopkeeper*, *Glenda*, *Mark*, *Rita*, *23*, and *21*, as well as the four nested relation instances representing first and last names of parents. In this relation, first and last names are no longer accessible separately.

4. *The Extended Nested Relational Algebra*

In this section, we introduce the basic operators of the extended nested relational algebra and discuss why we made this choice. Some of them are defined in the same way as their flat counterparts; among the others will be an operator that will allow us to disregard the internal structure of values. For a good comprehension it is necessary for us to define all the basic operators in a rigorous manner. In particular, we do not leave any ambiguity as to the names of the attributes in the scheme of the resulting relations.

We start with the binary operators; their definitions are straightforward generalizations of the corresponding flat relational algebra operators:

Definition 4.1. *Set-Operators and Cartesian Product*

—Given two nested relations (Ω, ω_1) and (Ω, ω_2). The *union* and the *difference* are defined by

$$(\Omega, \omega_1) \cup (\Omega, \omega_2) = (\Omega, \omega_1 \cup \omega_2),$$

$$(\Omega, \omega_1) - (\Omega, \omega_2) = (\Omega, \omega_1 - \omega_2).$$

—Given two nested relations (Ω_1, ω_1) and (Ω_2, ω_2) with $\text{ato}(\Omega_1) \cap \text{ato}(\Omega_2) = \varnothing$. The *Cartesian product* $(\Omega_1, \omega_1) \times (\Omega_2, \omega_2)$ is defined as $(\Omega_1 \cup \Omega_2, \omega')$ where

$$\omega' = \{t \in \mathscr{T}_{\Omega_1 \cup \Omega_2} \mid t|_{\Omega_1} \in \omega_1 \ \& \ t|_{\Omega_2} \in \omega_2\}.$$

As for the flat relational algebra, the *intersection* is not defined as a basic operator, since it can be expressed in terms of the difference. Note that, as in the flat case, we do not allow the Cartesian product of two nested relations the schemes of which share some atomic attributes. However, one has an intuitive notion about Cartesian products of relations that says that such operations can always be performed, no matter what the relations look like. What is obviously needed is a renaming operator as well as a notion of "isomorphic" nested relations and relation schemes. We shall come back to this issue later.

All the other operators that we introduce in this section are unary. For the precedence of operators, we adopt the same convention as for the flat relational algebra. The first unary operator that we introduce is also straightforwardly borrowed from the flat relational algebra:

Definition 4.2. *Projection.* Given a relation (Ω, ω) and $\Omega' \subseteq \Omega$. The *projection* $\pi_{\Omega'}(\Omega, \omega)$ equals (Ω', ω') where:

$$\omega' = \{t|_{\Omega'} \mid t \in \omega\}.$$

Next, we introduce as in [26] the nested relational algebra operators that are indispensable for any algebra dealing with nested relations: nesting and unnesting.

Definition 4.3. *Nesting and Unnesting*

—Given a relation (Ω, ω) and $X \subseteq \Omega$. The *nesting* $\nu_X(\Omega, \omega)$ equals (Ω', ω') where

$$\Omega' = (\Omega - X) \cup \{X\}$$

and, if $\Omega \neq X$, then

$$\omega' = \{t \in \mathscr{T}_{\Omega'} \mid \exists t' \in \omega : t|_{\Omega - X} = t'|_{\Omega - X} \ \& \ t(X) = \{t''|_X \mid t'' \in \omega \ \& \ t|_{\Omega - X} = t''|_{\Omega - X}\}\}$$

else, if $\Omega = X$, then

$$\omega' = \{\omega\}$$

—Given a relation (Ω, ω) and $X \in \Omega - U$. The *unnesting* $\mu_X(\Omega, \omega)$ equals (Ω', ω') where

$$\Omega' = (\Omega - \{X\}) \cup X$$

$$\omega' = \{t \in \mathcal{T}_{\Omega'} \mid \exists t' \in \omega : t|_{\Omega-\{X\}} = t'|_{\Omega-\{X\}} \;\&\; t|_X \in t'(X)\}$$

Example 4.1. Reconsider the relation in Example 2.1. If this relation is subsequently nested over the sets {CHILD, C-AGE}, {JOB}, {P-FNAME, P-LNAME} and {{P-FNAME, P-LNAME}, P-AGE, {JOB}}, then the result corresponds to the first tuple of the relation in Example 3.1. By unnesting the latter relation in the opposite order, we get back the relation of Example 2.1.

Note that in the classical definition (e.g., [26]), nesting is always defined as in the first case of our definition, also if $\Omega = X$. In this classical definition, nesting a relation (Ω, ω) over the entire scheme Ω always yields a singleton relation that from now on we denote as $(\{\Omega\}, \{\omega\})$, except for when $\omega = \emptyset$. Indeed, according to the definition of [26], the result of nesting $(\Omega, \emptyset)$ over its entire scheme is $(\{\Omega\}, \emptyset)$, which is an empty instance over a singleton scheme. In our definition, we wanted to avoid this "exception" by assuring that $\nu_\Omega(\Omega, \emptyset) = (\{\Omega\}, \{\emptyset\})$. This modification with respect to the classical definition will turn out very useful in the sequel.

We now define the copying operator that will, among other things, make it possible to disregard the internal structure of data.

Definition 4.4. *Copying.* Given a relation (Ω, ω), $X \in \Omega$ and $A \in U - \text{ato}(\Omega)$. The *copying* $\kappa_{A \leftarrow X}(\Omega, \omega)$ equals $(\Omega \cup \{A\}, \omega')$ where

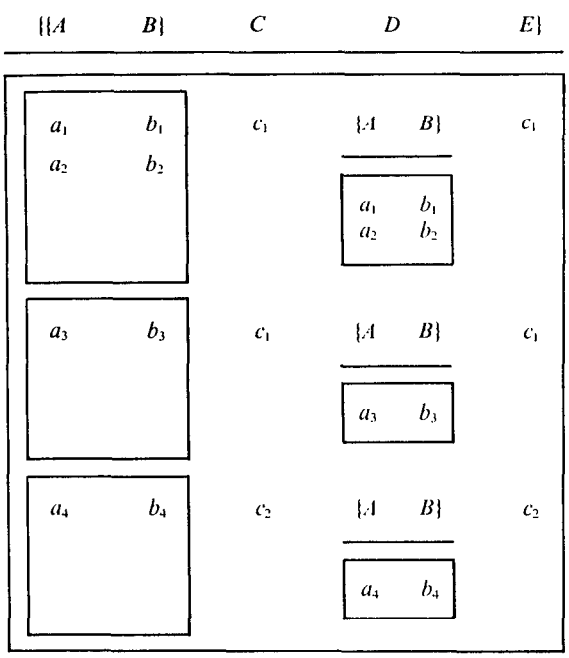
$$\omega' = \{t \in \mathcal{T}_{\Omega \cup \{A\}} \mid \exists t' \in \omega : t|_\Omega = t'|_\Omega \;\&\; t(A) = t'(X)\}.$$

So, the copying operator actually "copies" a column of the relation under consideration and uses a "new" atomic attribute as header of the copied column. We illustrate the use of this operator by an example.

Example 4.2. Let $A, B, C, D, E \in U$ and let (Ω, ω) be the following (abstract) relation:

{{A	B}	C}
a_1	b_1	c_1
a_2	b_2	
a_3	b_3	c_1
a_4	b_4	c_2

The result of $\kappa_{E \leftarrow C} \kappa_{D \leftarrow \{A,B\}}(\Omega, \omega)$ is



From now on, we call the operators defined in Definitions 4.1, 4.2, 4.3, and 4.4 the *basic operators* of the extended nested relational algebra. An *algebraic expression* is now formally defined as follows:

Definition 4.5

(1) For all $i > 0$, x_i is an algebraic expression;
(2) $(\varnothing, \varnothing)$ is an algebraic expression;
(3) For all algebraic expressions, the basic operators applied to these expressions, are also algebraic expressions.

The x_i in the above definition are called the *variables* of the algebraic expression (ae). If $E(x_1, \ldots, x_n)$ is an ae with variables $x_1, \ldots, x_n$ and $r_1, \ldots, r_n$ are relations, then $E(r_1, \ldots, r_n)$ is supposed to have the usual meaning (in particular, it is supposed to be undefined if this substitution would entail illegal operations). Two expressions $E_1(x_1, \ldots, x_n)$ and $E_2(x_1, \ldots, x_n)$ are said to be *equivalent* if for all relations $r_1, \ldots, r_n$, $E_1(r_1, \ldots, r_n)$ and $E_2(r_1, \ldots, r_n)$ are either both undefined or equal.

From Definition 4.5, it follows that we consider $(\varnothing, \varnothing)$, the empty instance over the empty scheme, as a "given" relation, which is a reasonable assumption. Indeed, as soon as we are given an arbitrary "input" relation, it is possible to construct $(\varnothing, \varnothing)$ as the projection of this relation onto the empty set of attributes.

All other operations on relations we encounter in the sequel are expressible as aes. If we compare our extended nested algebra with, for example, the one of [26], we see that the main difference between these two formalisms is in the presence of the copying operator in the former. In the remainder of this section, we show how the copying operator enables us to express various queries as aes. In this way we want to establish the strength of the copying operator as a basic operator, and

demonstrate that its introduction leads to a very uniform framework to query-nested relations.

A first possible application of the copying operator is its use in connection with the projection. Indeed, if we project out the column that was copied, then the net result is a *renaming*. For reasons of convenience, we explicitly define renaming as a derived operator.

Definition 4.6. Given a relation (Ω, ω), $X \in \Omega$ and $A \in U - \text{ato}(\Omega)$. The *renaming* $\rho_{A \leftarrow X}(\Omega, \omega)$ is defined by:

$$\rho_{A \leftarrow X}(\Omega, \omega) = \pi_{\Omega - \{X\}} \kappa_{A \leftarrow X}(\Omega, \omega).$$

Hence, the renaming is equivalent with an ae. Notice that Definition 4.6 encapsulates two notions of renaming attributes. In the first case, if X is an atomic attribute (i.e., $X \in U$), then $\rho_{A \leftarrow X}$ merely changes the name of X into A. We call this a *trivial renaming*; it is a straightforward generalization of the flat renaming operator in Definition 2.2. In the other case, where X is a nonatomic attribute, $\rho_{A \leftarrow X}$ is far from trivial; indeed, as a consequence of such a renaming, all the values originally corresponding to X by Definition 3.6 become prime; they become inaccessible, because A is an atomic attribute. Hence nontrivial renaming is the operator announced earlier that allows to disregard the internal representation of data.

Example 4.3. Reconsider the relations in Examples 3.1 and 3.5, respectively. The latter can be obtained from the former by the nontrivial renaming:

$$\rho_{\text{PARENT} \leftarrow \{\text{P-FNAME}, \text{P-LNAME}\}}.$$

Other good examples of data to which nontrivial renamings might be applied are addresses and phone numbers.

Of course, once the database manager has decided to disregard the internal structure of certain data by a nontrivial renaming, there is no way back!

Nontrivial renaming can also be used to create empty relations over arbitrary schemes.

PROPOSITION 4.1. *Let Ω be an arbitrary set of attributes. Then there exists an ae that expresses the relation $(\Omega, \varnothing)$.*

Rather than giving a formal proof of this proposition, we illustrate the ideas behind the proof by an example.

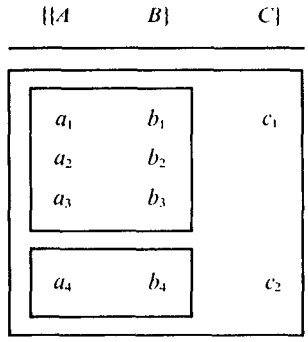
Example 4.4. Let $\Omega = \{\{A, B\}, C\}$ with $A, B, C \in U$. From Definition 4.5, it follows that $(\varnothing, \varnothing)$ is an ae. By the remarks following Definition 4.3, it follows that $\nu_{\varnothing}(\varnothing, \varnothing) = (\{\varnothing\}, \{\varnothing\})$. The difference of this latter relation and itself then yields $(\{\varnothing\}, \varnothing)$. By performing nontrivial renamings, we can then obtain the relations $(\{A\}, \varnothing)$, $(\{B\}, \varnothing)$, and $(\{C\}, \varnothing)$. By making a Cartesian product between the former two of these, we obtain $(\{A, B\}, \varnothing)$, nesting this relation over $\{A, B\}$ yields $(\{\{A, B\}\}, \{\varnothing\})$. The Cartesian product of this relation with $(\{C\}, \varnothing)$ finally gives us the desired relation $(\Omega, \varnothing)$.

Another important application of the copying operator is its use in connection with nesting and unnesting. From the literature (e.g., [28]), we recall the following result, already suggested by Example 4.1.

PROPOSITION 4.2. *Let (Ω, ω) be an arbitrary relation and let $X \subseteq \Omega$. Then $\mu_X(\nu_X(\Omega, \omega)) = (\Omega, \omega)$.*

So, any nesting can always be undone by the corresponding unnesting. The opposite, however, is generally *not* true, as is illustrated by the following example.

Example 4.5. Reconsider the relation (Ω, ω) of Example 4.2. The result of $\nu_{\{A,B\}}\mu_{\{A,B\}}(\Omega, \omega)$ is



Obviously, $\nu_{\{A,B\}}\mu_{\{A,B\}}(\Omega, \omega) \neq (\Omega, \omega)$.

The fact that, in general, an unnest cannot be undone by a nest is very unfortunate for the following reason. All the basic operators only involve the attributes at the highest level of the scheme under consideration. The only way to perform operators at a lower level is by first doing the appropriate unnestings, until we can "reach" the attributes we need to apply the above-defined operations, and after this, do some nestings to "restore" the structure of the original nested relation. Since unnestings in general cannot be undone by nestings, we need to introduce some additional information during the unnesting phase that will be used to restructure the original nested relation correctly. We introduce two operators that will help us to do this. The first one is defined in Definition 4.7. The existence of equivalent aes for these operators depends heavily on the use of the copying operator.

Definition 4.7. Let (Ω, ω) be a nested relation, let $X \in \Omega - U$ and let $A \in U - \text{ato}(\Omega)$. Then the *tagged unnesting* $\theta_{A \leftarrow X}(\Omega, \omega)$ is defined by

$$\theta_{A \leftarrow X}(\Omega, \omega) = \mu_X \kappa_{A \leftarrow X}(\Omega, \omega).$$

This operator satisfies the following property [28]:

PROPOSITION 4.3. *Let (Ω, ω) be a nested relation, let $X \in \Omega - U$ and let $A \in U - ato(\Omega)$. Let $\omega' = \{t \in \omega \mid t(X) \neq \emptyset\}$. Then*

$$(\Omega, \omega') = \pi_\Omega \nu_X \theta_{A \leftarrow X}(\Omega, \omega).$$

As an immediate corollary, we have

COROLLARY 4.1. *Let (Ω, ω) be a relation, let $X \in \Omega - U$ and let $A \in U - ato(\Omega)$. Suppose that $\forall t \in \omega : t(X) \neq \emptyset$. Then*

$$(\Omega, \omega) = \pi_\Omega \nu_X \theta_{A \leftarrow X}(\Omega, \omega).$$

So, intuitively, the tagged unnesting operator associates a "tag" with each tuple that enables us to reconstruct the tuple after unnesting, through nesting followed by projection.[4] In the case that in no tuple of the relation an empty value occurs for the nonatomic attribute that will be unnested, no information is lost. In the

[4] This technique has been used in several proofs in the literature (e.g., [28]).

following example, we show how this technique can be used to perform operations on a lower level.

Example 4.6. Reconsider the nested relation of Example 3.1. We recall the scheme of this relation (Ω, ω)

$$\Omega = \{\{\text{P-FNAME, P-LNAME}\}, \text{P-AGE}, \{\text{JOB}\}\}, \{\text{CHILD, C-AGE}\}\}.$$

Suppose that for some reason, we are no longer interested in the children's ages. We cannot "project out" this attribute directly, since C-AGE is not an attribute of the scheme of the relation. Now let C-DATA be an atomic attribute and consider

$$\theta_{\text{C-DATA}\leftarrow\{\text{CHILD,C-AGE}\}}.$$

The result of this operation is a nested relation (Ω', ω') with as scheme:

$$\Omega = \{\{\text{P-FNAME, P-LNAME}\}, \text{P-AGE}, \{\text{JOB}\}\}, \text{CHILD, C-AGE, C-DATA}\}.$$

The instance ω' of this nested relation could be represented as shown in Table IV.

We can now project out C-AGE (which is now an attribute of the scheme Ω') and nest over {CHILD}. The "tag" C-DATA will make sure that everything is grouped together in the same way as in the original relation.[5] After projecting out C-DATA, we obtain a nested relation (Ω'', ω'') with as scheme:

$$\Omega'' = \{\{\text{P-FNAME, P-LNAME}\}, \text{P-AGE}, \{\text{JOB}\}\}, \{\text{CHILD}\}\}.$$

The instance ω'' of this nested relation could be represented as shown in Table V.

Obviously, this is not quite yet the desired result. The second tuple of the original nested relation (Ω, ω) got "lost" when the θ-operator was applied to it. The reason for this is that the second tuple represents a couple without children, that is, has the empty relation as a value for the nonatomic attribute {CHILD, C-AGE} that was involved in the "tagging."

We now discuss a technique for not losing this information. Clearly, if tuples do occur that contain the empty set as a value for the attribute on which the tagged unnesting is performed, we need a second operator to keep track of them.

Definition 4.8. Let (Ω, ω) be a relation and let $X \in \Omega - U$. Let $\omega'' = \{t \in \omega \mid t(X) = \varnothing\}$. Then the *incomplete projection* $\phi_X(\Omega, \omega)$ is defined by

$$\phi_X(\Omega, \omega) = \pi_{\Omega-\{X\}}(\Omega, \omega'').$$

PROPOSITION 4.4. *The incomplete projection is equivalent to an ae.*

PROOF. Using Proposition 4.3 one can verify that for an arbitrary $A \in U - \text{ato}(\Omega)$:

$$\phi_X(\Omega, \omega) = \pi_{\Omega-\{X\}}((\Omega, \omega) - \pi_\Omega \nu_X \theta_{A\leftarrow X}(\Omega, \omega)). \qquad \square$$

[5] Note that in this particular example, we did not need to "tag" in order to be able to "recover" the "structure" of the original relation. This is so, because of the functional relationship between couples of parents and their set of children. Without this functional dependency present, "tagging" would have been unavoidable. We invite the reader to convince himself or herself of this by using the same technique for projecting out the attribute B in the relation (Ω, ω) of Example 4.2.

TABLE IV

{{{P-FNAME	P-LNAME}	P-AGE	{JOB }}	CHILD	C-AGE	C-DATA }
Jeff	Willows	54	professor manager	Glenda	23	{CHILD C-AGE} Glenda 23 Mark 21 Rita 21
Mary	Higgins	49	secretary			
Jeff	Willows	54	professor manager	Mark	21	{CHILD C-AGE} Glenda 23 Mark 21 Rita 21
Mary	Higgins	49	secretary			
Jeff	Willows	54	professor manager	Rita	21	{CHILD C-AGE} Glenda 23 Mark 21 Rita 21
Mary	Higgins	49	secretary			

TABLE V

{{{P-FNAME	P-LNAME}	P-AGE	{JOB }}	{CHILD }}
Jeff	Willows	54	professor manager	Glenda Mark Rita
Mary	Higgins	49	secretary	

By combining Proposition 4.3 and Definition 4.8, we get

PROPOSITION 4.5. *Let* (Ω, ω) *be a relation, let* $X \in \Omega - U$ *and let* $A \in U - ato(\Omega)$. *Then*

$$(\Omega, \omega) = \pi_{\Omega}\nu_X\theta_{A \leftarrow X}(\Omega, \omega) \cup \phi_X(\Omega, \omega) \times \nu_X(X, \varnothing).$$

By Proposition 4.2 and 4.4, the expression above is an ae. Note also that in this expression, we take full advantage of our modified definition of nesting (Definition 4.3).

Example 4.7. Reconsider Example 4.6. In this example, we wanted to "project out" the attribute C-AGE out of the relation (Ω, ω) of Example 3.1. Unfortunately, in trying to do so, we lost the second tuple, because it had the empty relation as a value for the nonatomic attribute {CHILD, C-AGE}. A correct answer to the query we asked, is

$$\pi_{\{\{\text{P-FNAME,P-LNAME}\},\text{P-AGE},\{\text{JOB}\}\},\{\text{CHILD}\}}\nu_{\{\text{CHILD}\}}$$
$$\pi_{\{\{\text{P-FNAME,P-LNAME}\},\text{P-AGE},\{\text{JOB}\}\},\text{CHILD,C-DATA}}\theta_{\text{C-DATA} \leftarrow \{\text{CHILD,C-AGE}\}}(\Omega, \omega)$$
$$\cup\ \phi_{\{\text{CHILD,C-AGE}\}}(\Omega, \omega) \times \nu_{\{\text{CHILD}\}}(\text{CHILD}, \varnothing).$$

Using a similar technique, it is also possible to perform other operations on lower levels with tagged unnesting and incomplete projection, even if in some tuples some nonatomic attributes are empty-valued for the attributes under consideration. In particular, one can perform trivial renamings on a lower level.

Definition 4.9. Let (Ω, ω) be a relation scheme. Let φ be a permutation on U. Extend φ to $\mathscr{U}$, $\mathscr{T}$, and $\mathscr{I}$ in the following way:

(1) If $X = \{X_1, \ldots, X_n\} \in \mathscr{U}$, then $\varphi(X) = \{\varphi(X_1), \ldots, \varphi(X_n)\}$;
(2) If $X \in \mathscr{U} - U$ and $t \in \mathscr{T}_X$, then $\varphi(t) \in \mathscr{T}_{\varphi(X)}$ and

$$\forall Y \in X: \varphi(t)(\varphi(Y)) = \begin{cases} t(Y) & \text{if } Y \in U; \\ \varphi(t(Y)) & \text{if } Y \notin U. \end{cases}$$

(3) If $X \in \mathscr{U} - U$ and $\omega \in \mathscr{I}_X$, then $\varphi(\omega) \in \mathscr{I}_{\varphi(X)}$ and $\varphi(\omega) = \{\varphi(t) \mid t \in \omega\}$.

Then the *global renaming* $\rho^{\varphi}(\Omega, \omega)$ is defined by

$$\rho^{\varphi}(\Omega, \omega) = (\varphi(\Omega), \varphi(\omega)).$$

PROPOSITION 4.6. *Any global renaming is equivalent to some ae.*

Global renaming makes it possible to "simulate" the intuitive notion of a Cartesian product between two arbitrary relations by altering the schemes in such a way that they have no longer any atomic attributes in common. We come back to this in the following section.

Finally one might feel that there is still one operator missing: the *selection.* Surprisingly enough, the selection can also be simulated by an ae, using the copying operator! As mentioned in the beginning of this article, the only properties of atomic values we consider are their equality or inequality. Here, we only consider equality selection; inequality selection can be defined in terms of equality selection and difference.

Definition 4.10. Given a relation (Ω, ω) and $A, B \in \Omega \cap U$. Then the *equality selection* (or, selection, for short) $\sigma_{A=B}(\Omega, \omega)$ equals (Ω, ω') where:

$$\omega' = \{t \in \omega \mid t(A) = t(B)\}.$$

PROPOSITION 4.7. *The selection is equivalent to an ae.*

PROOF. It is easily seen that, given a relation (Ω, ω):

$$\sigma_{A=B}(\Omega, \omega) = (\Omega, \omega) \cap \kappa_{B \leftarrow A} \pi_{\Omega - \{B\}}(\Omega, \omega). \qquad \square$$

Notice that the selection in Definition 4.10 is only defined between atomic attributes. Although at first glance this may seem an unreasonable restriction, it has no sense to define selection between nonatomic attributes. Indeed, we invite the reader to check that, by our definitions, nested values corresponding to different nonatomic attributes (i.e., nested relation instances with different schemes) are necessarily distinct. Nevertheless, it is possible to define the following "generalized" selection operator:

Definition 4.11. Let (Ω, ω) be a relation scheme. Let φ be a permutation on U (which is extended to $\mathscr{U}$, $\mathscr{T}$, and $\mathscr{I}$ as in Definition 4.9) such that $\varphi(\Omega) = \Omega$. Then the *global selection* $\sigma^{\varphi}(\Omega, \omega)$ equals (Ω, ω') where

$$\omega' = \{t \in \omega \mid \varphi(t) = t\}.$$

PROPOSITION 4.8. *Any global selection is equivalent to some ae.*

PROOF. First, observe that any global selection can be written as a chain of "primitive" selections that compare only one pair of attributes at a time. A primitive selection that compares a pair of atomic attributes is obviously an equality selection (Definition 4.10). Hence it suffices to show that primitive selections comparing nonatomic attributes can be written as aes.

Therefore, let (Ω, ω) be a relation scheme, let $X, Y \in \Omega - U$ and let φ be a permutation on U (extended to $\mathscr{U}$ and $\mathscr{F}$) such that $\varphi(\Omega) = \Omega$, $Y = \varphi(X)$, $X = \varphi(Y)$, and φ is the identity on $\Omega - \{X, Y\}$. Clearly, all primitive selections can be written as global selections of this type.

Now let φ' be another permutation on U (extended to $\mathscr{U}$ and $\mathscr{F}$) whose restriction to $\text{ato}(\Omega - \{X\})$ is the identity and let $\varphi'(X) = Z$. We furthermore assume that $\varphi'(Z) = X$ and $\text{ato}(\Omega) \cap \text{ato}(Z) = \varnothing$. Such a permutation φ' can always be constructed. Let us now explain why we need Z. Obviously, we will have to make "copies" of X and Y, rename them to some atomic attributes A and B, respectively, not in $\text{ato}(\Omega)$, and then compare them. However, the nested values associated with A would then be relation instances over the scheme X, whereas the values associated with B would be relation instances over the scheme Y and hence the former values would be incomparable to the latter ones. In order to avoid this problem, we rename X globally into some set Z, then copy Z in A, rename Z globally back into X, and repeat the same procedure for Y and B. Now, the nested values associated to A and B will both be relation instances over Z and hence be comparable. Then we can select for $A = B$ and project out these auxiliary attributes to obtain the desired result. For short,

$$\sigma^{\varphi}(\Omega, \omega) = \pi_{\Omega}\sigma_{A=B}\rho^{\varphi\circ\varphi'}\kappa_{B\leftarrow Z}\rho^{\varphi'\circ\varphi}\rho^{\varphi'}\kappa_{A\leftarrow Z}\rho^{\varphi'}(\Omega, \omega). \qquad \square$$

One might propose to write the primitive selection considered in the proof of Proposition 4.8 as $\sigma_{X=Y}$. Unfortunately, this notation is ambiguous, since it does not specify how the atomic attributes in X and Y, respectively, should be matched. (A similar argument holds for "primitive" global renamings.) However, whenever in a practical application there is no ambiguity as to how the matching should be done, we use this less rigorous but intuitively more appealing notation.

The remainder of this article will be devoted to two main issues. The first one might already have been hinted at by some constructions made in this section. On several occasions, we needed to consider permutations on U to express intuitive notions formally. This idea leads to the introduction of *isomorphic* relations and is developed in the following section. The other issue is concerned with the expressiveness of the extended nested relational algebra. This will be discussed in Section 6.

5. *Isomorphic Relations and Databases*

In the previous section, we observed that, in general, computing the Cartesian product of two arbitrary relation schemes, required some renamings. The validity of this strategy, that is, the result of it being meaningful, depends heavily on our philosophy of not attaching any meaning to attribute names. In other words, attribute names are merely used to distinguish the various entries of a tuple. In order to be consistent with this philosophy, we should work with classes of relations (or databases) that have the same "meaning," rather than with individual relations (or databases). In this section, this idea will be developed mainly for single relations. This is not really a restriction, since we show in the next section that any database can be represented by a single relation.

First though, we have to define classes of relations in such a way that a class contains relations with the same meaning. Therefore, we need to formally define the concept of "same meaning." In view of the remarks made earlier, it seems natural to use the following definition for that purpose:

Definition 5.1. Let (Ω_1, ω_1) and (Ω_2, ω_2) be two nested relations. These relations are said to be *isomorphic* if there exists a permutation φ on U such that $\varphi(\Omega_1) = \Omega_2$ and $\varphi(\omega_1) = \omega_2$ (where φ is extended to $\mathscr{U}$ and $\mathscr{F}$).

The isomorphism defined above is clearly an equivalence relation on the set of all nested relations. We call an element of the partition induced by this equivalence relation a *relation class.* The relation class defined by the nested relation r will be denoted by iso(r). Relations in the same relation class—and only those—are interpreted in our philosophy to have exactly the same meaning. Note that, apart from trivial variations on the attributes not belonging to the schemes under consideration, there might be several permutations establishing an isomorphism between two relations, as is illustrated below.

Example 5.1. Consider the following two relations:

$\{A$	B	$C\}$
a	b	c
b	a	c

$\{A$	B	$D\}$
a	b	c
b	a	c

Obviously, these relations are isomorphic. Any permutation on U for which A and B are fixpoints that maps C to D will do the job. Note however that all the permutations mapping A to B, B to A, and C to D are equally suited!

Of course, in order to justify the name "isomorphism," we need to show that the basic operators defined on relations in the previous section, can be extended to relation classes in a natural way. First, we consider the set operators, that is, union and difference. In the previous section, set operators were only defined on relations with the same scheme. The various ways one could think of to extend this requirement for relation classes turn out to be equivalent:

PROPOSITION 5.1. *Let $\mathscr{R}_1$ and $\mathscr{R}_2$ be two relation classes. The following statements are equivalent:*

(1) *There exists a relation $r_1 = (\Omega_1, \omega_1)$ in $\mathscr{R}_1$, a relation $r_2 = (\Omega_2, \omega_2)$ in $\mathscr{R}_2$, and a permutation φ on U such that $\Omega_2 = \varphi(\Omega_1)$;*
(2) *There exists a relation $r_1 \in \mathscr{R}_1$ and a relation $r_2 \in \mathscr{R}_2$ that have the same scheme;*
(3) *For each relation $r_1 \in \mathscr{R}_2$ there exists a relation $r_2 \in \mathscr{R}_2$ that has the same scheme as r_1;*
(4) *For each relation $r_1 = (\Omega_1, \omega_1)$ in $\mathscr{R}_1$ and each relation $r_2 = (\Omega_2, \omega_2)$ in $\mathscr{R}_2$ there exists a permutation φ on U such that $\Omega_2 = \varphi(\Omega_1)$.*

The proof of Proposition 5.1 is straightforward and, therefore, left to the reader. Two relation classes satisfying either one of (and hence all) the criteria of Proposition 5.1 are called *compatible.* Clearly, it makes only sense to define the set operators on compatible relation classes. As a first naive attempt, we might think that we can define the result of a set operator on two compatible relation classes

as the class of the relation obtained by applying the same set operator on two representing relations with the same scheme. This however does not work, not even for flat relations, as is shown in the following example:

Example 5.2. Consider two abstract relation classes, represented by the two relations below, respectively,

$r_1 =$

$\{A$	$B\}$
a	b
c	d

$r_2 =$

$\{A$	$B\}$
a	b
d	c

Both relation classes are represented by relations with the same scheme. $r_1 \cup r_2$ obviously equals

$r_1 \cup r_2 =$

$\{A$	$B\}$
a	b
c	d
d	c

Now consider the relation

$r_3 =$

$\{A$	$B\}$
b	a
c	d

r_2 and r_3 are equivalent (the attributes A and B are merely interchanged). If we take the union of r_1 and r_3, we get

$r_1 \cup r_3 =$

$\{A$	$B\}$
a	b
b	a
c	d

Clearly $r_1 \cup r_2$ is not equivalent to $r_1 \cup r_3$!

What we really need is a way of matching the "columns" of the relations in the first class with those of the relations in the second class. Attributes are not useful for this purpose since they are merely "headers" of "columns." In order to solve this problem, let us look at the fourth statement of Proposition 5.1. We know that any pair of relations, representing some given pair of compatible relation classes, can be linked together with some permutation on U. As can be seen from Example 5.1, this permutation is, in general, not unique. We can however establish a matching of "columns" by consistently selecting one particular permutation for each pair of relations.

Definition 5.2. Let $\mathscr{R}_1$ and $\mathscr{R}_2$ be two compatible relation classes. A *correspondence* between $\mathscr{R}_1$ and $\mathscr{R}_2$ is a mapping[6] $\chi: \mathscr{R}_1 \times \mathscr{R}_2 \rightarrow S(U)$ satisfying:

(1) If $r_1 = (\Omega_1, \omega_1)$ and $r_2 = (\Omega_2, \omega_2)$ and if $\chi(r_1, r_2) = \varphi$, then $\varphi(\Omega_1) = \Omega_2$;
(2) Let $r_1 = (\Omega_1, \omega_1)$ and $r_1' = (\Omega_1', \omega_1')$ be arbitrary relations of $\mathscr{R}_1$ and let $r_2 = (\Omega_2, \omega_2)$ and $r_2' = (\Omega_2', \omega_2')$ be arbitrary relations of $\mathscr{R}_2$. Suppose $\chi(r_1, r_2) = \varphi$ and $\chi(r_1', r_2') = \varphi'$. Let φ_1 be a permutation on U such that $\varphi_1(\Omega_1) = \Omega_1'$ and $\varphi_1(\omega_1) = \omega_1'$. Let $\varphi_2 = \varphi' \circ \varphi_1 \circ \varphi^{-1}$.[7] Then $\varphi_2(\Omega_2) = \Omega_2'$ and $\varphi_2(\omega_2) = \omega_2'$.

We invite the reader to check the consistency of this definition. The second condition of Definition 5.2 guarantees that the matching of "columns" is done in a consistent way. We can now define

Definition 5.3. *Set Operators.* Let $\mathscr{R}_1$ and $\mathscr{R}_2$ be compatible relation classes and let χ be a correspondence between them. Then the *union* and the *difference of* $\mathscr{R}_1$ *and* $\mathscr{R}_2$ *under* χ are defined by

$$\mathscr{R}_1 \overset{\chi}{\cup} \mathscr{R}_2 = \{\varphi(r_1) \cup r_2 \mid r_1 \in \mathscr{R}_1 \;\&\; r_2 \in \mathscr{R}_2 \;\&\; \chi(r_1, r_2) = \varphi\},$$

$$\mathscr{R}_1 \overset{\chi}{-} \mathscr{R}_2 = \{\varphi(r_1) - r_2 \mid r_1 \in \mathscr{R}_1 \;\&\; r_2 \in \mathscr{R}_2 \;\&\; \chi(r_1, r_2) = \varphi\}.$$

We leave it to the reader to check that the set defined above is indeed a relation class. To end this somewhat lengthy though necessary digression about the union (and the difference) of relation classes, we raise a practical notational issue.

Suppose r_1 and r_2 are two relations with the same scheme and suppose they represent relation classes. By Proposition 5.1, these relation classes are compatible. Since there may be several permutations establishing a connection between isomorphic relations, as shown in Example 5.1, there may also be several (not trivially equivalent) correspondences χ between $\text{iso}(r_1)$ and $\text{iso}(r_2)$ satisyfing the requirement that $\chi(r_1, r_2)$ is the identity on U. Nevertheless, the result of $\text{iso}(r_1) \cup^{\chi} \text{iso}(r_2)$ does not depend on the particular choice of such a χ; we leave it to the reader to check this. Hence, the expression $r_1 \cup r_2$ can be interpreted as the union between the classes $\text{iso}(r_1)$ and $\text{iso}(r_2)$ under some correspondence χ satisfying $\chi(r_1, r_2) = \text{id}_U$.

In contrast with the set operators, the Cartesian product of two relation classes is very easy to define

Definition 5.4. *Cartesian Product.* Let $\mathscr{R}_1$ and $\mathscr{R}_2$ be relation classes. Then the Cartesian product $\mathscr{R}_1 \times \mathscr{R}_2$ is defined as

$$\mathscr{R}_1 \times \mathscr{R}_2$$
$$= \{r_1 \times r_2 \mid r_1 = (\Omega_1, \omega_1) \in \mathscr{R}_1 \;\&\; r_2 = (\Omega_2, \omega_2) \in \mathscr{R}_2 \;\&\; \text{ato}(\Omega_1) \cap \text{ato}(\Omega_2) = \varnothing\}.$$

Again, it is straightforward to check that the result of the Cartesian product of two relation classes is again a relation class. Note that the Cartesian product makes sense for arbitrary relation classes since it is always possible to find representative relations the schemes of which have no atomic attributes in common. Hence, the expression $r_1 \times r_2$ now makes sense for all relations, provided this expression is interpreted as the Cartesian product of the classes these relations represent. This corresponds to our intuitive feeling already expressed in the previous section that the Cartesian product of arbitrary relations makes sense.

It still remains to extend the unary operators to relation classes. All the unary operators defined on relations in the previous section involve the specification of

[6] $S(U)$ stands for the set of all permutations on U.
[7] Note that the first equality always holds.

some attributes or sets of attributes. Therefore, if we work with relation classes, we have to specify, given a relation and a set of attributes of that relation, how to find the corresponding set of attributes in any other relation of the same class. Obviously, what we need to specify is a correspondence between a relation class and itself! Of course, we require that, in such a correspondence, each relation is associated with itself by the identical permutation on U.

Definition 5.5. Let $\mathscr{R}$ be a relation class. A *self-correspondence* χ on $\mathscr{R}$ is a correspondence χ between $\mathscr{R}$ and itself such that

$$\forall r \in \mathscr{R}: \chi(r, r) = \mathrm{id}_U.$$

In the remainder of this section, we assume to be given an arbitrary relation class $\mathscr{R}$ and an arbitrary self-correspondence χ on $\mathscr{R}$. In the context of such a relation class and self-correspondence, it is possible to extend the notion of an attribute.

Definition 5.6. An *attribute class* ξ is a mapping from $\mathscr{R}$ in $\mathscr{U}$ that satisfies

$$\forall r, r' \in \mathscr{R}: \chi(r, r') = \varphi \Rightarrow \varphi(\xi(r)) = \xi(r').$$

The notion of attribute class obviously depends on the relation class and self-correspondence under consideration. For simplicity's sake, however, this fact will not be mentioned explicitly in subsequent definitions, nor will it be reflected in the notation.

Several properties of attributes can be carried over to attribute classes. Indeed, let ξ be an attribute class and suppose, for example, that for some $r = (\Omega, \omega) \in \mathscr{R}$, $\xi(r) \in \Omega$. Then, clearly, this property holds for all r in $\mathscr{R}$. Therefore, such an attribute class is said *to belong to* $\mathscr{R}$. An analogous argument shows that it makes sense to speak of *atomic* and *nonatomic attribute classes*. Similarly, we can extend the ato-operator to attribute classes. With this knowledge, we can extend the unary operators to relation classes.

Definition 5.7. *Projection.* Let $\xi = \{\xi_1, \ldots, \xi_k\}$ be a set of attribute classes that belong to $\mathscr{R}$. Then the *projection* $\pi_\xi(\mathscr{R})$ is defined as

$$\pi_\xi(\mathscr{R}) = \{\pi_X(r) \mid r \in \mathscr{R} \;\&\; X = \{\xi_1(r), \ldots, \xi_k(r)\}\}.$$

Definition 5.8. *Nesting and Unnesting*

—Let $\xi = \{\xi_1, \ldots, \xi_k\}$ be a set of attribute classes that belong to $\mathscr{R}$. Then the *nesting* $\nu_\xi(\mathscr{R})$ is defined as

$$\nu_\xi(\mathscr{R}) = \{\nu_X(r) \mid r \in \mathscr{R} \;\&\; X = \{\xi_1(r), \ldots, \xi_k(r)\}\}.$$

—Let ξ be a nonatomic attribute class that belongs to $\mathscr{R}$. Then the *unnesting* $\mu_\xi(\mathscr{R})$ is defined as

$$\mu_\xi(\mathscr{R}) = \{\mu_X(r) \mid r \in \mathscr{R} \;\&\; X = \xi(r)\}.$$

Definition 5.9. *Copying.* Let ξ be an attribute class that belongs to $\mathscr{R}$, and let α be an atomic attribute class not in $\mathrm{ato}(\mathscr{R})$. Then the *copying* $\kappa_{\alpha \leftarrow \xi}(\mathscr{R})$ is defined as

$$\kappa_{\alpha \leftarrow \xi}(\mathscr{R}) = \{\kappa_{A \leftarrow X}(r) \mid r \in \mathscr{R} \;\&\; \xi(r) = X \;\&\; \alpha(r) = A\}.$$

We leave it to the reader to check that projection, nest, unnest, and copying are well defined, that is, that their results are again relation classes.

We now show that projections, nestings, and unnestings of relations can alternatively be interpreted as operations on relation classes. Therefore, consider, for

example, the expression $\pi_X(r)$. If r is interpreted as representing a relation class, and if some self-correspondence χ on this class is defined, then X can be seen as representing the attribute class ξ satisfying $\xi(r) = X$. Then $\pi_X(r)$ can be interpreted as a relation class, by Definition 5.8. Note furthermore that the result does not depend upon the particular choice of the self-correspondence. Of course, a similar argument holds for the nest and unnest operators and for the copying.

So, we now have dealt with all the basic operators defined in Section 4. We can summarize our results as follows:

(1) It is possible to extend all basic operators on relations to relation classes;
(2) It is possible to use expressions whose operands are relations, to denote the operations on the classes they represent *in an unambiguous way.*

Furthermore, we also want to make a comment on the function val defined in Section 3. Since obviously all relations of the same relation class contain the same prime values, val can alternatively be seen as a function on relation classes, rather than relations. We use this observation in the following section.

What have we gained by defining operators on relation classes rather than relations? First and foremost, we have a more natural framework for considering these operations, since attributes are merely column headers and hence relations that can be obtained from each other by applying a permutation to their scheme, are equivalent. We even have to work with relation classes, if we want the Cartesian product to be an operator that is always defined in an unambiguous manner. Moreover, working with relation classes implicitly—something many authors have done without ever saying—is no longer justified in the context of the extended nested algebra, considering that renaming is no longer a trivial operator.

We conclude this section with a short digression to *database classes.* We define them as follows:

Definition 5.10. Two databases are *isomorphic* if there exists a one-to-one mapping from the former database onto the latter such that corresponding relations are isomorphic.

Database classes are the elements of the partition induced by this equivalence relation.

Alternatively, one might think that a database class may also be defined as a set of relation classes. This, however, is *not* true, as shown by the following example:

Example 5.3. Consider a database of an electricity company, consisting of two relations. The first one contains for each customer the total amount due since the beginning of the year, whereas the second one shows the total amount paid since the beginning of the year. The scheme of the first relation might be

{CUSTOMER, ADDRESS, AMOUNT-DUE},

whereas the scheme of the second relation could be

{CUSTOMER, ADDRESS, AMOUNT-PAID}.

Observe that this database in Boyce–Codd normal form. Both of its relations contain customers, addresses, and amounts. Notice that, at the end of the year, when all balances must be settled, the amounts in both relations should (at least in theory) be equal. At that particular moment, both relations will be isomorphic. Nevertheless, these relations can still be distinguished; indeed, the names of the attributes in both relations can be compared, since they are contained in a single

database. Furthermore, it is very unlikely that these relations are isomorphic at any other time.

Definition 5.10 allows us to consider database classes representing a database such as the one we described in Example 5.3, above; if we would define a database class as a set of relation classes, this would not be possible. Note though that the relation classes determined by some representation of the database, do not depend on that particular representation. This observation will be used in the following section where we show that it suffices to look at single relations, instead of databases in general.

6. *The Expressiveness of the Extended Nested Relational Algebra*

From now on, while writing relations and databases, we mean the classes these relations and databases represent, unless explicitly stated otherwise. We invite the reader to check that all the definitions given below, even if they make use of specific representations of relation or database classes, can be interpreted as definitions about these classes in the obvious way.

In this section, we look at the expressiveness of the extended nested relational algebra. Therefore, we generalize the characterization of the basic information in Section 2. We use this characterization to show that our extended nested relational algebra is complete in the sense of Bancilhon and Paredaens [3, 4, 18]. The definition of *basic information of a nested relation or database* is entirely analogous to the corresponding definition for a flat relation or database (Definition 2.3), and will therefore not be repeated here.

As announced earlier, we now show why it suffices to consider single relations. Therefore we introduce the following notation:

Definition 6.1. Let d be a database. Suppose that d contains nonempty relations and let $r_1, \ldots, r_k$ be all these nonempty relations. Then $\Sigma(d) = r_1 \times \cdots \times r_k$. If d contains no nonempty relations, then $\Sigma(d) = (\varnothing, \varnothing)$.

THEOREM 6.1. *Let d be a database. Then $BI(d) = BI(\Sigma(d))$.*

PROOF. Obviously $\Sigma(d) \in \mathrm{BI}(d)$, by definition. Hence, $\mathrm{BI}(\Sigma(d)) \subseteq \mathrm{BI}(d)$. On the other hand, all nonempty relations in d can be obtained as some projection of $\Sigma(d)$. Furthermore, all empty relations in d can be expressed as aes, according to Proposition 4.1. It is now straightforward to check that $\mathrm{BI}(d) \subseteq \mathrm{BI}(\Sigma(d))$. □

As a consequence, it suffices to characterize the basic information of single relations only.

As for the basic information, we do not repeat the definition of *cogroup* for *nested relations* and *nested databases*, which is entirely analogous to Definition 2.4. Just note that the function val must be interpreted in the sense of Definition 3.6. Cogroups of nested relations were first considered in [28]. With the cogroup of a nested relation, it is straightforward to associate a flat relation in the following way:

Definition 6.2. Let r be a relation. Let τ be an arbitrary total one-to-one mapping from $\mathrm{val}(r)$ into U. Then the *cogroup relation of r* is defined by

$$\mathrm{CGR}(r) = (\tau(\mathrm{val}(r)), \omega'),$$

where

$$\omega' = \{\psi \circ \tau^{-1} |_{\tau(\mathrm{val}(r))} \mid \psi \in \mathrm{CG}(r)\}.$$

Notice that, in accordance with Definition 5.1, the cogroup relation does not depend on the particular choice of τ.

Example 6.1. Reconsider the relation in Example 3.1. In Example 3.6, we listed its set of prime values, which contains 21 elements. Clearly, the permutation ψ_1 on this set that interchanges *professor* and *manager* and keeps the other values fixed, leaves the relation invariant. The same holds for the permutation ψ_2 only interchanging *Mark* and *Rita*. We leave it to the reader to verify that the cogroup of the relation in Example 3.1 is generated by ψ_1 and ψ_2 and thus consists of a total of four permutations. Hence, its cogroup relation is a 4-tuple flat relation over a 21-attribute scheme, which can be represented as shown in Table VI.

If we define τ to be the mapping associating with each value the attribute corresponding to it in the first tuple, then, for example, the second tuple represents ψ_1 and the third ψ_2; the last tuple represents the composition of these permutations. (The first tuple of course represents the identical permutation.)

We invite the reader to determine the cogroup and the cogroup relation of the relation in Example 3.5.

Although all subsequent theorems deal with relations rather than databases, we have to consider databases and their cogroups in some of the proofs as intermediate constructs. It is straightforward to check that

PROPOSITION 6.1. *Let d be a database. Then $CG(d) = CG(\Sigma(d))$ and $CGR(d) = CGR(\Sigma(d))$.*

Obviously, the cogroup and the cogroup relation are merely two different ways of looking at the same notion. As explained earlier, we only consider equality and inequality among values and do not attach any further meaning to them (cf. also [3, 4, 18, 28]). Therefore it is reasonable to consider all permutations on the prime values of a relation that leave that relation invariant. Moreover, we claim that this set of permutations CG(r) contains all "information" about the relation r. If this is the case, then a relation and its cogroup relation must have the same basic information and cogroup. The former statement is proved later; the latter one is shown below:

THEOREM 6.2. *Let r be a relation. Then $CG(CGR(r)) = CG(r)$.*

PROOF. Let τ be an arbitrary total one-to-one mapping from val(r) into U, as in Definition 6.2.

$CG(CGR(r)) \supseteq CG(r)$. Let $\varphi \in \mathrm{CG}(r)$. Let $t = \psi \circ \tau^{-1}|_{\tau(\mathrm{val}(r))}$ be an arbitrary tuple of CGR(r) for some $\psi \in \mathrm{CG}(r)$. Then $\varphi(t) = \varphi \circ (\psi \circ \tau^{-1})|_{\tau(\mathrm{val}(r))} = (\varphi \circ \psi \circ \tau^{-1})|_{\tau(\mathrm{val}(r))}$. Since $\varphi \circ \psi \in \mathrm{CG}(r)$, it follows by Definition 6.2 that $\varphi(t) \in \mathrm{CGR}(r)$, whence $\varphi \in \mathrm{CG}(\mathrm{CGR}(r))$.

$CG(CGR(r)) \subseteq CG(r)$. Now let $\varphi \in \mathrm{CG}(\mathrm{CGR}(r))$. Then, by Definition 6.2, $\varphi(\psi \circ \tau^{-1}|_{\tau(\mathrm{val}(r))}) = \varphi \circ \psi \circ \tau^{-1}|_{\tau(\mathrm{val}(r))} \in \mathrm{CGR}(r)$. In particular, if ψ is the identical permutation on val(r), we have that $\varphi \circ \tau^{-1}|_{\tau(\mathrm{val}(r))} \in \mathrm{CGR}(r)$, whence, by Definition 6.2, $\varphi \in \mathrm{CG}(r)$. □

Now, recall Theorem 2.1 that gives a criterion for the expressiveness of the flat algebra. In [28], Theorem 2.1 was generalized to be the (classical) nested relational

TABLE VI

{A	B	C	D	E	F	G	H	I	J	K	L	M	N	P	Q	R	S	T	U	V}
Jeff	Mary	Tom	Brenda	Willows	Higgins	Jenkins	Jones	54	49	32	31	professor	manager	secretary	shopkeeper	Glenda	Mark	Rita	23	21
Jeff	Mary	Tom	Brenda	Willows	Higgins	Jenkins	Jones	54	49	32	31	manager	professor	secretary	shopkeeper	Glenda	Mark	Rita	23	21
Jeff	Mary	Tom	Brenda	Willows	Higgins	Jenkins	Jones	54	49	32	31	professor	manager	secretary	shopkeeper	Glenda	Rita	Mark	23	21
Jeff	Mary	Tom	Brenda	Willows	Higgins	Jenkins	Jones	54	49	32	31	manager	professor	secretary	shopkeeper	Glenda	Rita	Mark	23	21

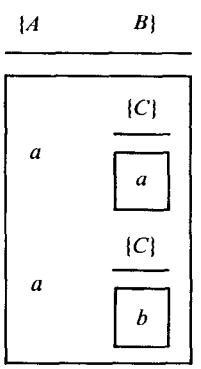
algebra (i.e., the obvious generalizations of the flat operators plus nesting and unnesting). However, Van Gucht [28] did not consider the possibility of changing the structure of a relation by nontrivial renaming. A generalization of Theorem 2.1 cannot hold without any modification for the extended nested relational algebra, since aes containing the copying operator can introduce new prime values in a relation. Hence, the relationship between the sets of prime values and between the cogroups of the relations we want to compare, cannot be a simple inclusion. However, these new prime values are actually nested relation instances composed of "older" prime values. More formally, if r is a nested relation and E is an ae such that $E(r)$ is well defined, then $\mathrm{val}(E(r)) \subseteq \mathrm{gen}(\mathrm{val}(r))$ (cf. Definitions 3.4 and 3.6). So we could hope to "save" the inclusion between the cogroups by extending all permutations on $\mathrm{val}(r)$ to $\mathrm{gen}(\mathrm{val}(r))$ in the natural way:

Definition 6.3. Let r be a relation and let $\psi \in \mathrm{CG}(r)$. The *extension of* ψ to $\mathrm{gen}(\mathrm{val}(r))$ is a binary relation R on $\mathrm{gen}(\mathrm{val}(r))$ defined by:

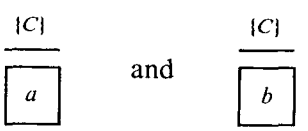
(1) For all $v \in \mathrm{val}(r)$: $vR\psi(v)$;
(2) If $\omega \in \mathscr{I}_\Omega$ is in $\mathrm{gen}(\mathrm{val}(r))$, then $\omega R\omega'$ if $\omega' \in \mathscr{I}_\Omega$ and if there is a bijection ϕ from the tuples of ω to the tuples of ω' such that for all $t \in \omega$ and for all $X \in \Omega$, $t(X)R\phi(t)(X)$.

Unfortunately, this naive approach does not work, since these extensions are not always functions, let alone permutations, as is shown by the following counterexample.

Example 6.2. Let A, B, $C \in U$ and a, b, $\in V$. Consider the following relation r:

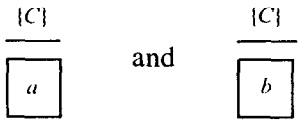


Since B is an atomic attribute,



are prime values of r. The permutation on $\mathrm{val}(r)$ that interchanges these prime values and leaves a and b fixed is obviously in $\mathrm{CG}(r)$. The natural extension of this

permutation to gen(val(r)) is no longer a function, since, if it were, it should leave



fixed on the one hand and interchange them on the other hand, a contradiction.

In order to solve the problem raised above, we need to make an observation that is essential for the sequel. If a relation contains nonatomic prime values, then it is reasonable to assume that these values were originally associated to nonatomic attributes that were subsequently renamed to atomic attributes by a decision of the dbms-manager. So, we might impose the condition that in the original relation, of which we want to compute and/or describe the basic information, only atomic values are associated to atomic attributes. This condition, however, is too restrictive, since it does not allow the recursive approach to the theorems we need to prove in the sequel. It turns out that it suffices to impose the following restriction:

Definition 6.4. A nested relation r is called *basic* if each permutation $\psi \in \mathrm{CG}(r)$ can be extended in the natural way to a permutation on gen(val(r)) (cf. Definitions 3.4 and 3.6). A nested database d is called *basic* if each permutation $\psi \in \mathrm{CG}(d)$ can be extended in the natural way to a permutation on gen(val(d)).

In particular, we have (as should be expected)

PROPOSITION 6.2. *Let r be a relation such that $\mathrm{val}(r) \subseteq V$. Then r is basic.*

PROPOSITION 6.3. *A database d is basic if and only if the relation $\Sigma(d)$ is basic.*

Furthermore, we have

PROPOSITION 6.4. *Let r be a basic relation. Then $\mathrm{CGR}(r)$ is basic.*

PROOF. Follows from Theorem 6.2 and the fact that, by definition, $\mathrm{val}(r) = \mathrm{val}(\mathrm{CGR}(r))$. □

As mentioned before, we intend to compare the cogroup of a relation with the cogroup of the result of the application of an ae to that relation. In order to do so, we introduce the following notation:

Definition 6.5. Let Ψ_1 and Ψ_2 be sets of permutations on finite sets $\mathscr{A}_1$ and $\mathscr{A}_2$, respectively, with $\mathscr{A}_1 \subseteq \mathscr{V}$ and $\mathscr{A}_2 \subseteq \mathrm{gen}(\mathscr{A}_1)$. Suppose that each permutation of Ψ_1 is extendible to a permutation on $\mathrm{gen}(\mathscr{A}_1)$ in the natural way. Then $\Psi_1 \sqsubseteq \Psi_2$ means that the restriction to $\mathscr{A}_2$ of the extension of each permutation of Ψ_1 to $\mathrm{gen}(\mathscr{A}_1)$ is in Ψ_2.

Using the notation introduced above, Theorem 2.1 can be rephrased in the context of the nested relational model as follows:

LEMMA 6.1. *Let r and s be nested relations. If r and s are both flat (i.e., if they have flat schemes), then there exists an ae $E(x)$ using only union (Definition* 4.1), *difference (Definition* 4.1), *Cartesian product (Definition* 4.1),

projection (*Definition* 4.2), *trivial renaming* (*Definition* 4.6), *and equality selection* (*Definition* 4.10) *such that* $s = E(r)$ *if and only if* $val(s) \subseteq val(r)$ *and* $CG(r) \sqsubseteq CG(s)$.

PROOF. Let U be the set of all atomic attributes and V the set of all atomic values in the nested relational model in which r and s are defined. Now consider the flat relational model (Section 2) of which the set of all attributes is U and the set of all values contains V as well as all prime values of r and s. Obviously, r and s are flat relations in this model. Theorem 2.1 now immediately yields the desired result. □

LEMMA 6.2. *Let* r *be a basic relation. Let* s, s_1, s_2 *be relations such that* $CG(r) \sqsubseteq CG(s)$, $CG(r) \sqsubseteq CG(s_1)$, *and* $CG(r) \sqsubseteq CG(s_2)$. *Whenever the following operations are properly defined, we have*

—$CG(r) \sqsubseteq CG(s_1 \cup s_2)$;
—$CG(r) \sqsubseteq CG(s_1 - s_2)$;
—$CG(r) \sqsubseteq CG(s_1 \times s_2)$;
—$CG(r) \sqsubseteq CG(\pi_{\Omega'}(s))$;
—$CG(r) \sqsubseteq CG(\nu_X(s))$;
—$CG(r) \sqsubseteq CG(\mu_X(s))$;
—$CG(r) \sqsubseteq CG(\kappa_{A \leftarrow X}(s))$.

PROOF. Straightforward. □

If $r = s$, some statements of Lemma 6.2 can be sharpened.

COROLLARY 6.1. *Let* r *be a relation. Let* A *and* B *be atomic attributes and suppose that all operations below are well defined. Then, using the notation introduced in Theorem* 2.1, *we have*:

—$CG(r)|_{val(\pi_{\Omega'}(r))} \subseteq CG(\pi_{\Omega'}(r))$;
—$CG(r) = CG(\nu_X(r))$;
—$CG(r)|_{val(\mu_X(r))} \subseteq CG(\mu_X(r))$;
—$CG(r) = CG(\kappa_{B \leftarrow A}(r))$;
—$CG(r) = CG(\rho_{B \leftarrow A}(r))$;
—$CG(r)|_{val(\sigma_{A=B}(r))} \subseteq CG(\sigma_{A=B}(r))$.

We may conclude from Lemma 6.2

THEOREM 6.3. *Let* r *be a basic relation and let* $s \in BI(r)$. *Then* $CG(r) \sqsubseteq CG(s)$.

The remainder of this section will be devoted to showing that the converse of Theorem 6.3 holds as well. To do so, we take maximum advantage of the previously established Theorem 2.1 for the flat relational model, by using Lemma 6.1.

THEOREM 6.4. *Let* r *be an arbitrary relation. There exists a flat relation* r' *such that* $BI(r') = BI(r)$.

PROOF. The construction of r' out of r using aes is an inductive process. Consider a database d that is initialized as $d_0 = \{r\}$. Of course, $\mathrm{BI}(d_0) = \mathrm{BI}(r)$. The

value of d after the ith step of the construction will be denoted d_i. Now let $d_i = \{r_1, \ldots, r_n\}$ and assume that $\mathrm{BI}(d_i) = \mathrm{BI}(r)$. For each nonflat relation $r_j = (\Omega_j, \omega_j) \in d_i$, let $X_j \in \Omega_j - U$ be an arbitrary nonatomic attribute of r_j and let $A_j \in U - \mathrm{ato}(r_j)$ be an atomic attribute not occurring in r_j. d_{i+1} is constructed out of d_i by replacing each nonflat relation $r_j \in d_i$ by the two relations $\theta_{A_j \leftarrow X_j}(r_j)$ and $\phi_{X_j}(r_j)$. By Proposition 4.5 and the induction hypothesis, $\mathrm{BI}(d_{i+1}) = \mathrm{BI}(d_i) = \mathrm{BI}(r)$. Obviously, this process will end. Let d' be the resulting value of d and let $r' = \Sigma(d')$. Since d' is obviously flat, so is r'. By Theorem 6.1 and by induction, $\mathrm{BI}(r') = \mathrm{BI}(d') = \mathrm{BI}(r)$. □

We illustrate the construction in the proof of Theorem 6.4 by an example.

Example 6.3. Consider the following nested relation:

$r =$

$\{A$	$\{B\}$	$\{C\}\}$
a_1		
a_2		c_2
a_3	b_3	c_3 c_4

First, let $d_0 = \{r\}$ and let $X = \{C\}$. Then $d_1 = \{r_1, r_2\}$ with

$r_1 =$

$\{A$	$\{B\}$	C	$D\}$
a_2		c_2	$\{C\}$ c_2
a_3	b_3	c_3	$\{C\}$ c_3 c_4
a_3	b_3	c_4	$\{C\}$ c_3 c_4

$r_2 =$

$\{A$	$\{B\}\}$
a_1	

Now let $X_1 = \{B\}$ and $X_2 = \{B\}$. We then obtain $d_2 = \{r_{11}, r_{12}, r_{21}, r_{22}\}$ with

$r_{11} =$

$\{A$	B	C	D	$E\}$
a_3	b_3	c_3	$\{C\}$: c_3, c_4	$\{B\}$: b_3
a_3	b_3	c_4	$\{C\}$: c_3, c_4	$\{B\}$: b_3

$r_{12} =$

$\{A$	C	$D\}$
a_2	c_2	$\{C\}$: c_2

$r_{21} =$

$\{A$	B	$F\}$

$r_{22} =$

$\{A\}$
a_1

Obviously, d_2 is flat, do $d' = d_2$. In accordance with Definition 5.4, $r' = \Sigma(d')$ equals

$r' =$

$\{A$	B	C	D	E	A'	C'	D'	$A''\}$
a_3	b_3	c_3	$\{C\}$: c_3, c_4	$\{B\}$: b_3	a_2	c_2	$\{C\}$: c_2	a_1
a_3	b_3	c_4	$\{C\}$: c_3, c_4	$\{B\}$: b_3	a_2	c_2	$\{C\}$: c_2	a_1

This nine attribute-flat relation has the same basic information as r.

We invite the reader to verify the following property:

PROPOSITION 6.5. *The relation r' obtained in the proof of Theorem* 6.4 *is unique, that is, does not depend on the particular choices in each step of the construction.*

It is therefore justified to denote r' henceforth as $\Psi(r)$. We do not attempt to give a proof of Proposition 6.5, since such a proof would be very long and technically involved and would deviate us from main goals of this article. Moreover, the uniqueness of $\Phi(r)$, though convenient, is not essential in the sequel.

In order to conclude our discussion on $\Phi(r)$, we wish to make a comment on the algebraic expression for $\Phi(r)$ constructed in the proof of Theorem 6.4. This expression does not only depend on the scheme, but also on the actual instance of

r. However, a verification of Definition 6.1 and the proof of Theorem 6.4 reveals that the only instance-dependent part in the expression is the "translation" of "Σ" into an ae. Indeed, the number of nonempty relations in d' depends on the number of "empty values" in the instance of r. Other models for nested relations do not allow empty relation instances as nested values within a nested relation (e.g., [28]). If we would impose this restriction to the nested relational model as defined in Section 3, the ae for $\Phi(r)$ would no longer be instance-dependent and could hence be interpreted as a query, which might be convenient in applications. However, we do not impose such a restriction.

We now establish the following properties of $\Phi(r)$.

LEMMA 6.3. *Let r be a relation. Then $CGR(\Phi(r)) \in BI(r)$ and $r \in BI(CGR(\Phi(r)))$.*

PROOF. Since, by Theorem 6.2, $\mathrm{CG}(\mathrm{CGR}(\Phi(r))) = \mathrm{CG}(\Phi(r))$ and since $\mathrm{CGR}(\Phi(r))$ and $\Phi(r)$ and both flat relations, it follows from Lemma 6.1 that $\Phi(r) \in \mathrm{BI}(\mathrm{CGR}(\Phi(r)))$ and $\mathrm{CGR}(\Phi(r)) \in \mathrm{BI}(\Phi(r))$. Theorem 6.4 then yields the desired result. □

Next, we want to show that the function Φ always produces basic relations when applied to basic relations. Therefore, we need a technical lemma.

LEMMA 6.4. *Let $d = \{r_1, \ldots, r_n\}$ be a basic nonflat database. Assume that $r_1 = (\Omega, \omega)$ is a nonflat relation. Let $X \in \Omega - U$ be a nonatomic attribute of r_1 and let $A \in U - ato(\Omega)$ be an atomic attribute not occurring in r_1. Then $\bar{d} = \{\theta_{A \leftarrow X}(r_1), \phi_X(r_1), r_2, \ldots, r_n\}$ is a basic database.*

PROOF. Let $\psi \in \mathrm{CG}(\bar{d})$. We have to prove that ψ is extendible to a permutation on $\mathrm{gen}(\mathrm{val}(\bar{d}))$.

First, we show that the restriction $\psi|_{\mathrm{val}(d)}$ of ψ to $\mathrm{val}(d)$ is in $\mathrm{CG}(d)$. Obviously, $\psi|_{\mathrm{val}(d)}(r_i) = r_i$ for all $i = 2, \ldots, n$. We only have to prove that $\psi|_{\mathrm{val}(d)}(r_1) = r_1$. Therefore, let t be a tuple in r_1. We show $\psi|_{\mathrm{val}(d)}(t)$ is in r_1. We have to distinguish two cases.

Case 1: $t(X) = \emptyset$. Consider the restriction $t|_{\Omega - \{X\}}$ of t to $\Omega - \{X\}$. By definition, $t|_{\Omega-\{X\}}$ is in $\phi_X(r_1)$. Hence, by assumption, $\psi|_{\mathrm{val}(d)}(t|_{\Omega-\{X\}}) = \psi(t|_{\Omega-\{X\}})$ is in $\phi_X(r_1)$. Hence, by definition, there exists a tuple t' in r_1 with $t'(X) = \emptyset$ and $t'|_{\Omega-\{X\}} = \psi|_{\mathrm{val}(d)}(t|_{\Omega-\{X\}})$. Obviously, $\psi|_{\mathrm{val}(d)}(t) = t'$ is in r_1, as had to be shown.

Case 2: $t(X) \neq \emptyset$. Now let $\{t_1, \ldots, t_k\} = \{\bar{t} \in \mathcal{T}_{(\Omega-\{X\})\cup X\cup\{A\}} \mid \bar{t}|_{\Omega-\{X\}} = t|_{\Omega-\{X\}} \;\&\; \bar{t}|_X \in t(X) \;\&\; \bar{t}(A) = t(X)\}$. By definition, $t_1, \ldots, t_k$ are in $\theta_{A \leftarrow X}(r_1)$. Hence, by assumption, $\psi(t_1), \ldots, \psi(t_k)$ are in $\theta_{A \leftarrow X}(r_1)$. Since $\psi(t_1)(A) = \cdots = \psi(t_k)(A) = \psi(t(X))$, it follows by definition that there exists t' in r_1 such that $\{\psi(t_1), \ldots, \psi(t_k)\} \subseteq \{\bar{t} \in \mathcal{T}_{(\Omega-\{X\})\cup X\cup\{A\}} \mid \bar{t}|_{\Omega-\{X\}} = t'|_{\Omega-\{X\}} \;\&\; \bar{t}|_X \in t'(X) \;\&\; \bar{t}(A) = t'(X)\}$. It can be easily shown that the above inclusion is actually an equality. Hence, $\psi|_{\mathrm{val}(d)}(t) = \psi(t) = t'$ is in r_1, as had to be shown.

Since $\psi|_{\mathrm{val}(d)} \in \mathrm{CG}(d)$, we know by assumption that $\psi|_{\mathrm{val}(d)}$ can be extended to a permutation on $\mathrm{gen}(\mathrm{val}(d))$. Let ψ' be this extension. In order to show that ψ is extendible to a permutation on $\mathrm{gen}(\mathrm{val}(d))$, it now suffices to show that the restriction $\psi'|_{\mathrm{val}(\bar{d})}$ of ψ' to $\mathrm{val}(\bar{d})$ equals ψ. Clearly, all nested values of $\mathrm{val}(\bar{d}) - \mathrm{val}(d)$ occur in $\theta_{A \leftarrow X}(r_1)$ and are associated to the atomic attribute A in the latter relation. Therefore, let t_θ be a tuple in $\theta_{A \leftarrow X}(r_1)$ and consider $t_\theta(A)$. By

definition, there exists t in r_1 with $t(X) = t_\theta(A)$. Now let t' in r_1 be as in Case 2 above. There, we already showed that $t' = \psi|_{\mathrm{val}(d)}(t)$. Since $t(X) = t_\theta(A)$, it follows that $\psi'(t_\theta(A)) = \psi|_{\mathrm{val}(d)}(t(X)) = t'(X)$. On the other hand, it also follows from the reasoning in Case 2 that $\psi(t_\theta(A)) = t'(X)$. Hence, $\psi'(t_\theta(A)) = \psi(t_\theta(A))$, which completes the proof. □

We can now show

LEMMA 6.5. *Let r be a basic relation. Then* $\Phi(r)$ *is also a basic relation.*

PROOF. Let us review the construction of $\Phi(r)$ in the proof of Theorem 6.4. Obviously, $d_0 = \{r\}$ is a basic database. Each step in the construction can be decomposed into a number of finer steps in which only one relation is replaced at a time. By applying Lemma 6.4 inductively to each of these finer steps, we eventually obtain that the database d' in the proof of Theorem 6.4 is basic. Proposition 6.3 then yields that $\Phi(r)$ is basic. □

In the next theorem, we derive a result about the relationship between the cogroup relations of r and its flat counterpart $\Phi(r)$.

THEOREM 6.5. *Let r be a basic relation. Then* $CG(r) \sqsubseteq CG(\Phi(r))$ *and* $CG(\Phi(r)) \sqsubseteq CG(r)$.

PROOF. By Theorem 6.4, $\mathrm{BI}(r) = \mathrm{BI}(\Phi(r))$. Lemma 6.5 now allows us to apply Theorem 6.3 in both directions, whence Theorem 6.5. □

COROLLARY 6.2. *Let r be a basic relation. Then* $CGR(r) \in BI(r)$.

PROOF. Consider $\Phi(r)$. Clearly $\mathrm{val}(r) \subseteq \mathrm{val}(\Phi(r))$. Now let X be the set of attributes in the scheme of $\mathrm{CGR}(\Phi(r))$ corresponding to elements of $\mathrm{val}(r)$. Then, from Theorem 6.5, it follows that $\mathrm{CGR}(r) = \pi_X(\mathrm{CGR}(\Phi(r)))$. By Lemma 6.3, $\mathrm{CGR}(\Phi(r)) \in \mathrm{BI}(r)$. Hence, $\mathrm{CGR}(r) \in \mathrm{BI}(r)$. □

Corollary 6.2 corresponds to an intuition explained earlier that, if membership of the basic information could be decided by looking at cogroups, then the cogroup relation and the original relation should have the same basic information. Corollary 6.2 proves one inclusion of this claim. The other will be shown as a corollary to the main theorem of this section, Theorem 6.6. First though, we need a technical lemma in order to prove Theorem 6.6.

LEMMA 6.6. *Let r be a basic relation and let* $\Delta \subset gen(val(r))$ *be a finite set. Then there exists a flat basic relation* $r' \in BI(r)$ *with*:

—$\Delta \subset val(r')$;
—$CGR(\Phi(r))$ *is some projection of* r';
—$BI(r) = BI(r')$.

PROOF. We first note that the last condition of Lemma 6.6 follows from the second one, by Lemma 6.3. So we only show the first and second statement. The proof goes by induction. Assume that we have a basic flat relation $s \in \mathrm{BI}(r)$ satisfying the second condition of Lemma 6.6. (In the first step, we can take $s = \mathrm{CGR}(\Phi(r))$, by Lemmas 6.3 and 6.4 and Proposition 6.2.) Now let $\delta \in \Delta - \mathrm{val}(s)$. By Theorem 6.5, we also have that $\mathrm{CGR}(r)$ is a projection of s. Hence, for all $a, b \in \mathrm{val}(r)$, there exist different atomic attributes A and B in the scheme of S such that for some tuple t in s, $t(A) = a$ and $t(B) = b$. Hence, by a sequence of copyings involving only atomic attributes and nestings, it is possible to create a relation s' such that for some nonatomic attribute X in the scheme of s' and some

tuple t' in s', $t(X) = \delta$. By Corollary 6.1, $CG(s') = CG(s)$. Hence, s' is also a basic relation. Now consider $s'' = \Phi(s')$. By Theorem 6.4 and Lemmas 6.3 and 6.4, s'' is a basic flat relation satisfying the second (and the third) condition of Lemma 6.6. Furthermore, $val(s) \cup \{\delta\} \subseteq val(s'')$. Since Δ is finite, it is possible to repeat this process until also the first condition of Lemma 6.6 is satisfied. □

We can now state our main result.

THEOREM 6.6. *Let r be a basic relation and s an arbitrary relation. Then $s \in BI(r)$ if and only if $val(s) \subseteq gen(val(r))$ and $CG(r) \sqsubseteq CG(s)$.*

PROOF

only-if. If r is a nested relation, and $E(x)$ is an ae such that $E(r)$ makes sense, then obviously $val(E(r)) \subseteq gen(val(r))$. The only-if direction now follows immediately from Theorem 6.3.

if. Consider $\Phi(s)$. Obviously $val(\Phi(s)) \subseteq gen(val(r))$. Furthermore, by Theorems 6.3 and 6.4, $CG(r) \sqsubseteq CG(\Phi(s))$. Since $val(\Phi(s))$ is necessarily finite, we know there exists a basic flat relation r' satisfying the conditions of Lemma 6.6 with $\Delta = val(\Phi(s))$. Since, by Lemma 6.6, $BI(r) = BI(r')$, it follows from Theorem 6.3 that $CG(r') \sqsubseteq CG(r)$. Hence $CG(r') \sqsubseteq CG(\Phi(s))$. Since r' and $\Phi(s)$ are both flat relations and since furthermore $val(\Phi(s)) \subseteq val(r')$ by construction, we may now conclude by Lemma 6.1 that $\Phi(s) \in BI(r') = BI(r)$. By Theorem 6.4, $s \in BI(r)$, which concludes the proof. □

COROLLARY 6.3. *Let r be a basic relation. Then $BI(CGR(r)) = BI(r)$.*

PROOF. By definition $val(CGR(r)) = val(r)$. By Theorem 6.1, $CG(CGR(r)) = CG(r)$. Hence, a twofold application of Theorem 6.6 yields $CGR(r) \in BI(r)$ and $r \in BI(CGR(r))$, whence Corollary 6.3. Note that the inclusion from left to right follows already from Corollary 6.2. □

So, even when nontrivial renamings are introduced, it is possible to characterize the basic information of a relation in terms of the set of permutations on atomic values that leave the relation invariant.

We now finally come back to an issue raised in the introduction: the completeness of the extended nested algebra in the sense of Bancilhon and Paredaens [3, 4, 18]. Unfortunately, the original definition of this notion cannot be applied straightforwardly to the extended nested algebra, since our operators can introduce new prime values in a relation. However, BP-completeness for the relational and the nested relational algebra is equivalent to a statement of which Theorem 6.6 is the most natural generalization for the extended nested relational algebra. Hence, it is justified to consider satisfaction of Theorem 6.6 as the definition of BP-completeness for query languages defined on the model introduced in Section 3. In this sense, the extended nested relational algebra is BP-complete.

7. *Conclusions*

We extended the nested relational algebra by adding the possibility of disregarding the internal structure of values by renaming. In doing so, we were forced to handle attribute names carefully, although they are only used to distinguish the "columns" in a relation, and were not given any semantical meaning. This made us realize we had to consider classes of semantically isomorphic relations rather than individual relations. We redefined the extended nested algebra operators for relation classes

and showed that algebraic expressions on individual relations can alternatively be interpreted as operations on the classes they represent, in an unambiguous way. For the extended nested algebra on classes of isomorphic relations, we were able to generalize the results of Paredaens [18] and Van Gucht [28] concerning the characterization of the basic information of a relation in terms of its cogroup. This result allowed us to conclude that the extended nested algebra is complete in the sense of Bancilhon and Paredaens [3, 4, 18].

REFERENCES

1. ABITEBOUL, S., AND BEERI, C. On the power of languages for the manipulation of complex objects. INRIA Tech. Rep. 846. INRIA, Paris, France, May 1988.
2. ABITEBOUL, S., AND BIDOIT, N. Non first normal form relations to represent hierarchically organized data. In *Proceedings of the 3rd Symposium on Principles of Database Systems* (Waterloo, Ont., Canada, Apr. 2–4). ACM, New York, 1984, pp. 191–200.
3. BANCILHON, F. On the completeness of query languages for relational data bases. In *Proceedings of the 7th Symposium on Mathematical Foundations of Composition Science* (Zakopane, Poland). In Lecture Notes in Computer Science, vol. 64, Springer-Verlag, Berlin, 1978, pp. 112–123.
4. CHANDRA, A. K., AND HAREL, D. Computable queries for relational databases. *J. Comput. Syst. Sci. 21*, 2 (Oct. 1980), 156–178.
5. CODD, E. F. A relational model of data for large shared data banks. *Commun. ACM 13*, 6 (June 1970), 377–387.
6. CODD, E. F. Further normalizations of the relational data base model. In *Data Base Systems*, R. Rustin, Ed. Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 33–64.
7. CODD, E. F. Relational completeness of database sublanguages. In *Data Base Systems*, R. Rustin, Ed. Prentice-Hall, Englewood Cliffs, N.J., 1972, pp. 65–98.
8. CODD, E. F. Extending the database relational model to capture more meaning. *ACM Trans. Datab. Syst. 4*, 4 (Dec. 1979), 397–434.
9. DESHPANDE, V., AND LARSON, P. An algebra for nested relational databases. Tech. Rep. CS-87-65. Univ. of Waterloo, Waterloo, Ont., Canada, Dec. 1987.
10. GYSSENS, M., AND VAN GUCHT, D. The powerset operator as an algebraic tool for understanding least fixpoint semantics in the context of nested relations. Tech. Rep. 233. Computer Science Department, Indiana Univ., Bloomington, Ind., Oct. 1987.
11. GYSSENS, M., AND VAN GUCHT, D. The powerset algebra as a result of adding programming constructs to the nested relational algebra. In *Proceedings of the ACM SIGMOD Conference on Management of Data* (Chicago, Ill., June 1–3). ACM, New York, 1988, pp. 225–232.
12. JAESHKE, G., AND SCHEK, H. J. Remarks on the algebra on non-first normal form relations. In *Proceedings of the 1st Symposium on Principles of Database Systems* (Los Angeles, Calif., Mar. 29–30). ACM, New York, 1982, pp. 124–138.
13. KORTH, H. F., AND SILBERSCHATZ, A. *Database System Concepts*. McGraw-Hill, New York, 1986.
14. KUPER, G. M., AND VARDI, M. Y. A new approach to database logic. In *Proceedings of the 3rd Symposium on Principles of Database Systems* (Waterloo, Ont., Canada, Apr. 2–4). ACM, New York, 1984, pp. 86–96.
15. MAIER, D. *The Theory of Relational Databases*. Computer Science Press, Rockville, Md., 1983.
16. MAKINOUCHI, A. A consideration of normal form of not-necessarily-normalized relations in the relational data model. In *Proceedings of the 3rd VLDB Conference* (Tokyo), 1977, pp. 447–453.
17. ÖZSOYOĞLU, G., ÖZSOYOĞLU, Z. M., AND MATOS, V. Extending relational algebra and relational calculus with set-valued attributes and aggregate functions. *ACM Trans. Datab. Syst. 12*, 4 (Dec. 1987), 566–592.
18. PAREDAENS, J. On the expressive power of the relational algebra. *Inf. Proc. Lett. 7*, 2 (Feb. 1978), 107–111.
19. PAREDAENS, J., DE BRA, P., GYSSENS, M., AND VAN GUCHT, D. The structure of the relational database model. In *EATCS Monographs on Theoretical Computer Science*. Vol. 17, Springer-Verlag, Berlin, 1989.
20. PAREDAENS, J., AND VAN GUCHT, D. Possibilities and limitations of using flat operators in nested algebra expressions. In *Proceedings of the 7th Symposium on Principles of Database Systems* (Austin, Tex., Mar. 21–23). ACM, New York, 1988, pp. 29–38.
21. PISTOR, P., AND TRAUNMUELLER, R. A database language for sets, lists and tables. In *Proceedings of the 12th VLDB* (Kyoto), 1986, pp. 278–288.

22. ROTH, M. A., KORTH, H. F., AND BATORY, D. S. SQL/NF: A query language for ¬1NF relational databases. *Inf. Syst. 12*, 1 (1987), 99–114.
23. ROTH, M. A., KORTH, H. F., AND SILBERSCHATZ, A. Null values in ¬1NF relational databases. Tech. Rep. TR-85-32, Univ. Texas, Austin, Tex., 1985.
24. ROTH, M. A., KORTH, H. F., AND SILBERSCHATZ, A. Extended algebra and calculus for ¬1NF relational databases. *ACM Trans. Datab. Syst. 13*, 4 (Dec. 1988), 389–417.
25. SCHEK, H.-J., AND SCHOLL, M. H. The relational model with relation-valued attributes. *Inf. Syst. 11*, 2 (1986), 137–147.
26. THOMAS, S. J., AND FISCHER, P. C. Nested relational structures. In *The Theory of Databases*, P. C. Kanellakis, Ed. JAI Press, Greenwich, Conn., 1986, pp. 269–307.
27. ULLMAN, J. D. *Principles of Database and Knowledge-Base Systems*, vol. 1. Computer Science Press, Rockville, Md., 1988.
28. VAN GUCHT, D. On the expressive power of the extended relational algebra for the unnormalized relational model. In *Proceedings of the 6th Symposium on Principles of Database Systems* (San Diego, Calif., Mar. 23–25). ACM, New York, 1987, pp. 302–312.
29. ZANIOLO, C. Database relations with null values. *J. Comput. Syst. Sci. 28*, 1 (Feb. 1984), 142–166.

RECEIVED AUGUST 1988; REVISED JANUARY 1989; ACCEPTED JANUARY 1989