# Efficient evaluation of relational queries using "nested relations"

Nicole Bidoit

INRIA

CENTRE DE ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
BP 105
78153 Le Chesnay Cedex
France

Tél. : (1) 39 63 55 11

Rapports de Recherche

N° 480

# EFFICIENT EVALUATION OF RELATIONAL QUERIES USING „NESTED RELATIONS"

Nicole BIDOIT

Janvier 1986

# Efficient Evaluation of Relational Queries Using "Nested Relations"

N. BIDOIT[1]

ABSTRACT

The purpose of this paper is to demonstrate that, for the Verso model, nesting relations not only provides a more convenient data representation but also leads to a more succint and efficient data manipulation. We exhibit a simple but highly expressive class of the relational queries whose expressions can be simulated by a Verso selection followed by a Verso projection. We proceed in a constructive manner using an extension of the tableau technique to represent relational expressions and characterize these expressions. The work presented here is a good basis for implementing an efficient relational interface for the Verso Data Base Machine.

RESUME

Ce papier a pour objet de démontrer que, dans le cadre du modèle V-relationnel, la structure de relation non-normalisée ne propose pas seulement une représentation des données plus agréable et succinte mais permet également de manipuler les données de façon plus efficace et succinte. Nous exhibons un sous ensemble simple de l'algèbre relationelle dont chaque expression peut être traduite par une Verso-selection suivie d'une Verso-projection. Nous proposons une caractérisation constructive de ces expressions relationnelles utilisant une extension de la technique des tableaux. Ce travail constitue également un étude en vue de la réalisation d'une interface relationnelle à la Machine Base de Données Verso.

[1]INRIA, Domaine de Voluceau, FRANCE. This research was done while visiting the Univeristy of Southern California, Los Angeles

## 1. Introduction

Since Makinouchi [Makinouchi 77] first investigated the idea of relaxing the first normal restriction [Codd 70] and of allowing set-valued domains, nested relations (also called non-first-normal form relations) have attracted a lot of attention, motivated by new applications of database systems ( [Macleod 83, Ozsoyoglu 83, Kobayashi 80, Bancilhon + Richard + Scholl 82, Bancilhon 82]). It has been enhanced that nesting relations leads to a more natural and succint data representation and the related theoretical aspects of non first normal relations are mainly contained in [Scheck + Pistor 82, Scheck + Scholl 84, Jaeschke + Scheck 82, Fisher + Thomas 83a, Fisher + Thomas 83b, Abiteboul + Bidoit 84a, Abiteboul + Bidoit 84b]. The purpose of this paper is to demonstrate that, for the Verso model [Abiteboul + Bidoit 84a], nesting relations not only provides a more convenient data representation but also leads to a more succint and efficient data manipulation.

In the Verso model [Abiteboul + Bidoit 84a] data is organized in non first normal form relations. The recursive definition of a Verso schema induces a hierarchical organization of data and the implicit specification of join dependencies. It also allows to represent some simple type of incomplete information. A simple and complete algebraic language allowing data restructuring as been defined for Verso relations. As mentionned earlier, the purpose of this paper is to demonstrate the expressive power of the Verso query language. As claimed in [Abiteboul + Bidoit 84a], the operations proposed on Verso instances take advantage of the semantic connection among attributs and some queries which would typically required joins in the pure relational model can be expressed by a

unique selection in the Verso model. Here, a simple but highly expressive class of relational queries that can be interpreted by simple Verso expressions involving at most two unary operations, namely projection and selection, is characterized and furthermore a technique for recognizing these queries is developped. The advantage of our approach is that preserving some positive features of the relational model, tools developped in the relational framework can be used in the Verso framework. Indeed, an extension of the tableau technique is used here to represent in a uniform way both a subclass of relational expressions and a subclass of Verso expressions.

We have to mention here that the Verso model is a formalization of some of the concepts used in the Verso database Machine [Bancilhon 82]. The data structure in the Verso database Machine is Verso instances stored sequentially and fully sorted. This storage allows fast access and processing [Bancilhon + Richard + Scholl 82]. Indeed all the operations of the Verso model but restructuring that requires sorting, are implemented by a specialized filter. In particular the Verso selection-projection expressions are converted into finite automatas runned by the Verso filter. This directly entails that all the relational queries characterized here to be expressable by a simple Verso selection-projection can be processed "on the fly" by the Verso filter. Also the technique associated with the characterization can be directly implemented to provide a relational user interface for data base systems using nested relations as internal data structure.

The paper is organized as follows. In the first section, we review some fundamental issues of the Verso model, in particular the correspondence between the Verso model and

the relational model. In the second section, we generalize the tableau technique to represent relational projection-selection-join expressions defined on particular schemas, called format skeletons. Modifications of classical results on tableau equivalence are obtained. In the third section, we study how to use these tableaux to characterize relational projection-selection-join expressions on format skeletons expressible by simple unary Verso expressions.

## 2. Preliminaries

We assume that the reader is familiar with the relational model [Ullman ]. In this section, we briefly review some basic concepts and present the main notations used throughout the paper.

We assume the existence of an infinite set of attributes U, and for each attribute A in U, of a set of values called the domain of A and denoted DOM(A). A relational schema is a finite set of attributes. Let V be a relational schema, then a tuple v over V is a mapping from V into $\cup_{A\in V}$DOM(A) such that v(A)$\in$DOM(A) for each A in V. The set of tuples over V is denoted by TUP(V). A (first normal form) relation over V is a finite set of tuples over V and the set of relations over V is denoted REL(V).

The relational operations of union, intersection, difference, join, projection and selection are respectively denoted by $\cup$, $\cap$, -, $\bowtie$, $\Pi_P$ (where P is the set of projected attributes) and $Select_C$ where C is a condition on a set V of attributes. An elementary condition on an attribute A is a disjunction of conditions of the form $A<a$, $A>a$, $A\leq a$, $A\geq a$, $A=a$, $A\neq a$ where $a\in$DOM(A). A condition on a set V of attributes is a conjunction of elementary conditions on attributes in V. If C is a condition on the attribute A (resp. on the set of attributes V) and $a\in$DOM(A) (resp. v is tuple over V), then a satisfies C (resp. v satisfies C) is denoted $a\models C$ (resp. $v\models C$). If $C_1$ and $C_2$ are conditions such that each value (or tuple) satisfying $C_1$ also satisfies $C_2$ then we say that $C_1$ implies $C_2$ denoted by $C_1\Rightarrow C_2$. For the purpose of our discussion, we introduce two supplementary conditions, namely the "empty" condition, denoted True, satisfied by all

values (or tuples) and the "full" condition, denoted False, satisfied by no value (no tuple).

A relational database schema is a finite set of relational schemas. A relational database instance r of some relational database schema R is a mapping from R into $\cup_{V \in R} REL(V)$ such that $r(V) \in REL(V)$ for each V in R. A relational database instance r satisfies the Universal Relation Schema Assumption (URSA) if $\Pi_X(r(Y)) \subseteq r(X)$ for each X,Y in R and $X \subseteq Y$. Let R be a database schema, then a relational projection-selection-join expression (PSJ-expression) on R and its target schema are defined recursively by:

1. [V] is a PSJ-expression on R for each V in R, (V is called the target schema of [V]),
2. Assume taht $E_1$, $E_2$ are two PSJ-expressions on R with target schemas $V_1$, $V_2$ respectively; that $Y \subseteq V_1$ and C is a condition on the set of attribute $V_1$; and that $V_1 \cap V_2 \neq \emptyset$. Then:
   a. $\Pi_Y[E_1]$ is a PSJ-expression on R (with target schema Y),
   b. $Select_C[E_1]$ is a PSJ-expression on R (with target schema $V_1$),
   c. $E_1 \bowtie E_2$ is a PSJ-expression on R (with target schema $V_1 \cup V_2$).

In general, we use the letters A, B,... to denote attributes; a, b,... to denote domain values; V, W, X, Y,... to denote relational schemas; v, w, x, y,... to denote tuples; R, S,... to denote relational database schemas and r, s,... to denote relational database instances. We also use the classical convention of writing XY for the union of two sets of attributes X and Y.

## 3. The Verso Model

In this section, we present the data structure of the Verso model and introduce the operations on Verso relations. We also include some results concerning the simple connection between Verso relations and relational database instances satisfying the URSA as well as the interpretation of the Verso operations in terms of relational operations. The reader interested in a complete description of the different aspects of the Verso model is invited to refer to [Abiteboul + Bidoit 84a] and [Bidoit 84] .

Let us first consider an example. A department consists of a set of COURSEs, the BOOKs required for each course, the STUDENTs registered in each course and their GRADEs. Figure 3-1 represents the information in a department. Intuitively, a department can be considered as a relation over, say, the three attributes, COURSE, $A_1$ and $A_2$. The values of the attribute COURSE are atomic whereas the values of $A_1$ and $A_2$ are simpler Verso relations. Note that in the example, there is no BOOK required for the "physics" COURSE. Thus null values can be represented in a Verso relation. Also note that an implicit connection is assumed between the attributes STUDENT and BOOK through the attribute COURSE. In other word, a "join" is implicit between COURSE STUDENT and COURSE BOOK.

To specify the structure of a Verso relation we use the auxiliary concept of a format. Informally, a <u>format</u> f is a regular expression of the form $X(f_1)^*...(f_n)^*$ where X is a finite set of attributes, $f_1,...,f_n$ are formats (possibly empty) and no attribut occurs twice in f. If Z is the set of attributes strictly occuring in the definition of a format f, we say that f

is a format on Z. For convenience, a directed tree representation for formats is used as shown in Figure 3-1 for the format COURSE(BOOK)*(STUDENT(GRADE)*)*. Verso relations are now formally defined in terms of format instances.

**Definition:** Let f be a format. The set of all instances over f, denoted Inst(f) ,is defined recursively as follows:

1. if f≡X then Inst(f)=REL(X).
2. if f≡X$(f_1)^*$...$(f_n)^*$ then I is in Inst(f) iff
   a. I is a finite subset of TUP(X)XInst($f_1$)X...XInst($f_n$), and
   b. if <u$I_1$...$I_n$> and <u$I_1'$...$I_n'$> are in Inst(f) for some u,$I_1$,...,$I_n$,$I_1'$,...,$I_n'$ then $I_i$=$I_i'$ for i=1..n.

Intuitively, the definition above states that an instance I takes atomic values on the attributes in X and not atomic but instance values on the "attributes" $f_1$,...,$f_n$. The definition also forces X to be a key. An instance of COURSE(BOOK)*(STUDENT(GRADE)*)* is represented in Figure 3-1.

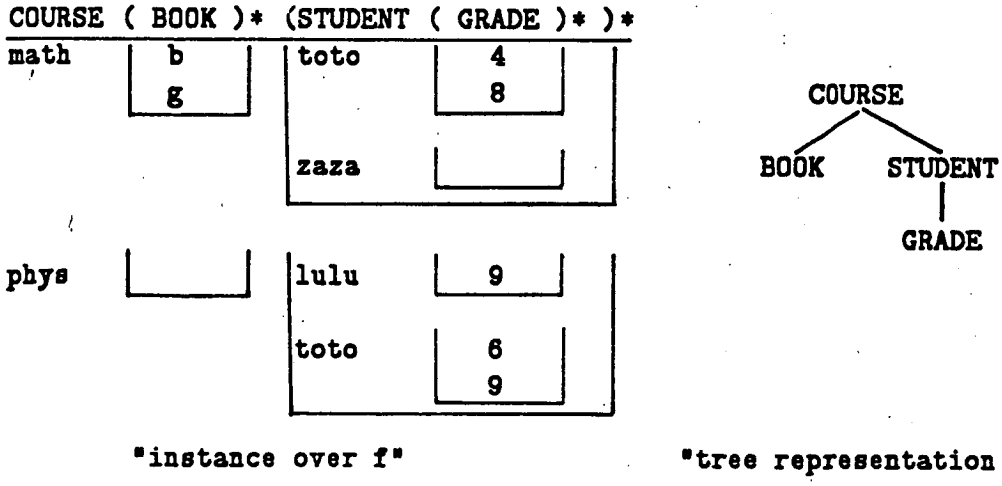


**Figure 3-1:** Verso relations are format instances.

In [Abiteboul + Bidoit 84b], we present a complete set of operations on Verso relations on the form of an algebra. The binary operation of the Verso algebra are defined noot only on Verso relations having same structure, but also on Verso relations having compatible structures. The notion of format compatibility is defined using the notion of a subformat. A format g is a subformat of a format f if the tree associated with g is a subtree of the tree assoociated with f and has same root. Intuitively, if g is a subformat of f and J an instance over g, an instance over f, called the extension of J on f and denoted by $J^f$, containing the same information as J, can be obtained simply by "padding" at each level with empty instances. An instance J over COURSE(STUDENT)* and its extension on COURSE(BOOK)*(STUDENT(GRADE)*)* are represented in Figure 3-2. Now, two formats f and g are compatible if they are both subformats of some third format h. This format h is the common structure on which instances over both f and g can be represented.

The binary Verso operations are union, adding the information contents of two Verso relations; intersection, extracting the information common to two Verso relations; difference, removing the information contained in a Verso relation from the information contained in a second one; join combining the information contents of two Verso relations having compatible structures (and intersecting the information contents of two Verso relations having same structure).

In [Abiteboul + Bidoit 84a], the binary operations are defined using an inclusion relation on Verso relations.

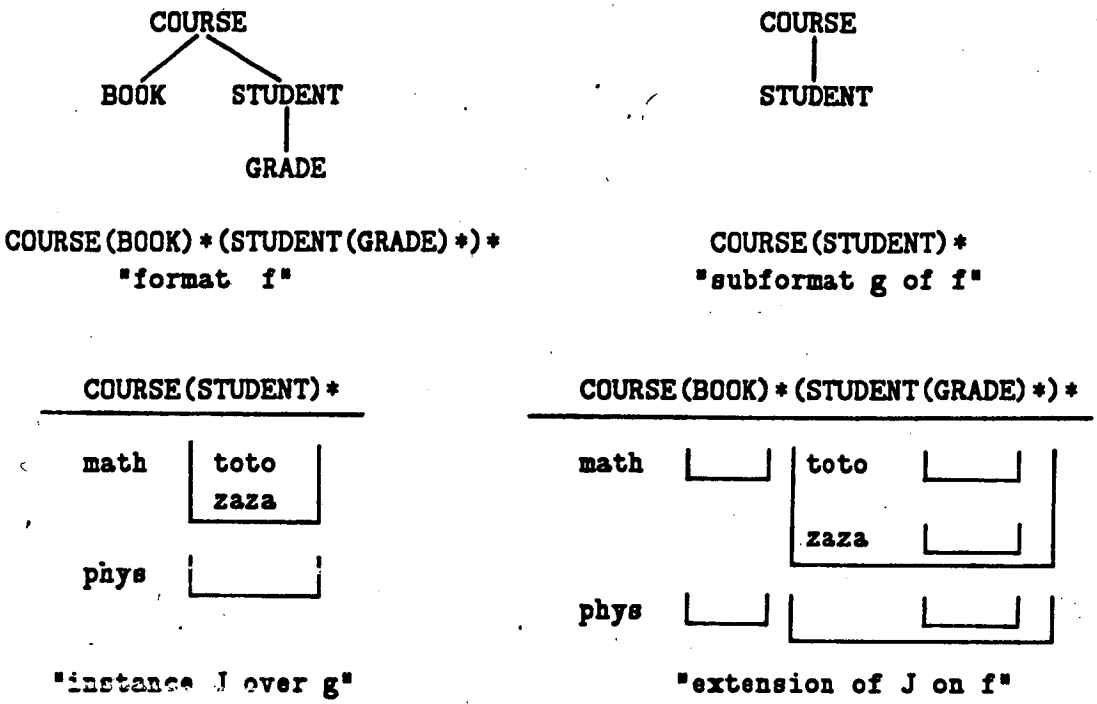


**Figure 3-2:** Subformat and Extension.

**Definition:** Let f be a format and I, J two instances over f. Then I is <u>included in</u> J, denoted $I \leq J$, iff:

if $f \equiv X(f_1)^*...(f_n)^*$ , $f_1,...,f_n$ not empty, then $\forall <uI_1...I_n> \in I$, $\exists <uJ_1...J_n> \in J \mid I_i \leq J_i$ for $i=1..n$.

We formally define the Verso union, and difference using the inclusion relation on Verso relations. The Verso intersection is not presented here as its definition is embeded in the Verso join definition.

**Definition:** Let f and g be two compatible formats and h a format such that f and g

are subformats of f. Let I and J be two instances over f and g respectively. Then, the union of I and J according to h, denoted $I \oplus_h J$ is the smallest instance defined over h containing $I^h$ and $J^h$; the difference of I and J according to h, denoted $I \ominus_h J$ is the smallest instance over h, included in $I^f$ such that its union with J according to h is equal to $I \oplus_f J$.

As a set operation definition of the Verso join requires the former definition of projection, a constructive definition of the Verso join is given below. In [Abiteboul + Bidoit 84a], the binary operations are presented in both algebraic and constructive fashion.

**Definition:** Let f and g be two compatible formats and h a format such that f and g are subformats of f. Let I and J be two instances over f and g respectively. Then the join of I and J according to h is defined recursively by:

1. if $h \equiv X$ then $(f \equiv g \equiv X)$ $I \circledast_h J = I \cap J$ and,
2. if $h \equiv X(h_1)^* \ldots (h_n)^*$, $f \equiv X(f_1)^* \ldots (f_n)^*$, $g \equiv X(g_1)^* \ldots (g_n)^*$ where for each $i=1..n$, $f_i$ ( respectively $g_i$) is a subformat of $h_i$, possibly empty. Then:

$$I \circledast_h J = \{ <uK_1 \ldots K_n> \mid <uI_1 \ldots I_n> \in I^h,\ <uJ_1 \ldots J_n> \in J_h,$$
$$K_k = I_k \circledast_{h_k} J_k \text{ if } f_k \text{ and } g_k \text{ are not empty,}$$
$$K_k = I_k \text{ if } f_k \text{ non empty, } g_k \text{ empty and,}$$
$$K_k = J_k \text{ if } f_k \text{ empty, } g_k \text{ not empty}\}$$

In Figure 3-3, the binary operations are illustrated using an instance I over $COURSE(STUDENT)^*$ and an instance J over $COURSE(BOOK)^*$, the common format used to express the result of the operations applied to I and J being $f \equiv COURSE(STUDENT)^*(BOOK)^*$.

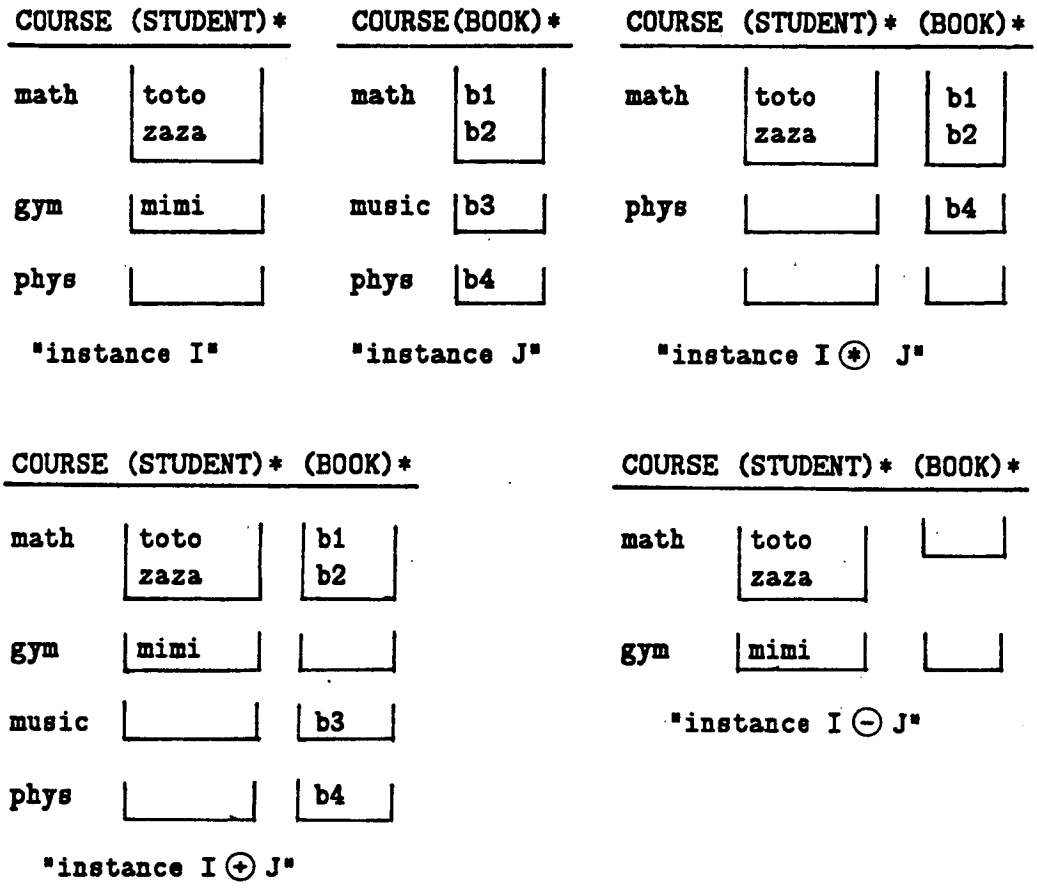


**Figure 3-3:** Binary Verso operations.

The unary operations of our model are extension (described above), projection, selection, restriction, renaming and restructuring. The last unary operation, namely restructuring, allows data reorganization and in [Abiteboul + Bidoit 84b] some key issues raised by this data restructuring have been emphasized. For the purpose of this paper, we focus here on the presentation of the two unary operations of projection and selection.

The Verso projection defined here is slightly more general than the Verso projection

over subformat presented in [Abiteboul + Bidoit 84a]. In order to present the Verso projection, we need the auxiliary concept of projected format. Intuitively, if f is a format on Z, and P a set of attributes included in Z, then the projection of f on P, denoted by $f_{|P}$, is obtained by removing from f all attributes not in P, as well as parentheses and star associated with empty strings.

**Example 3.1:** Let f≡COURSE(BOOK)*(STUDENT(GRADE)*)* be a format. Consider the three sets of attributes X={COURSE,STUDENT}, Y={STUDENT,GRADE} and Z={BOOK,GRADE}.

1. $f_{|X}$=COURSE(STUDENT)* is a format and a subformat of f.
2. $f_{|Y}$=STUDENT(GRADE)* is a format but not a subformat of f.
3. $f_{|Z}$=(BOOK)*(GRADE)* is not a format.

□

The preceding example shows that the projection of a format f on a set P of attributes is not always a format. The projection of an instance I over f on a set of attributes P is only defined when $f_{|P}$ is a format. The instance obtained is an instance over $f_{|P}$. The constraint on the set P preserves the attribute hiercharchy induced by the format f and leads to a simple and sound semantics for the Verso projection.

We give below the formal definition of the operation of Verso projection.

**Definition:** Let f be a format on Z, and P be a non-empty subset of Z such that $f_{|P}$ is a format. Let I be an instance over f. Then the projection of I on P, denoted $I[f_{|P}]$ (or I[P] when f is understood), is the instance over $f_{|P}$ recursively defined by:

1. if $f \equiv X$ then $I[P]=\{v \mid v \in Tup(P)$ and $\exists u \in I \mid \forall A \in P\ u(A)=v(A)\}$
2. if $f \equiv X(f_1)^*...(f_n)^*$, $f_1,...,f_n$ formats on $Y_1,...,Y_n$ respectively,

$$I[P]=\oplus_{<uI_1..I_n>\in I} \{<vJ_1...J_n> \mid v=u[X \cap P] \text{ and } J_j=I_i[Y_i \cap P] \text{ where } J_j \text{ is defined on } f_{i|Y_i \cap P},\ Y_i \cap P \neq \emptyset\}.$$

It is clear that the definition above generalizes the definition of the relational projection. In Figure 3-4, we present an instance I over COURSE(BOOK)*(STUDENT(GRADE)*)* and its projection on {STUDENT,GRADE}.

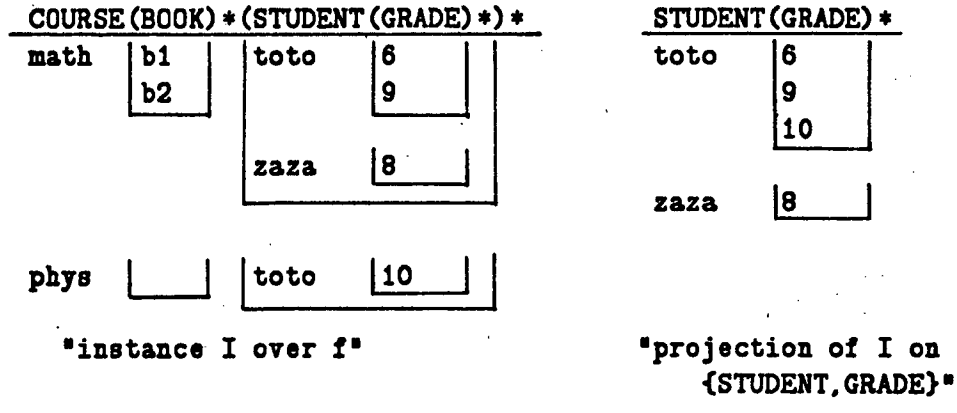


**Figure 3-4:** Verso projection.

The second important unary operation of our model is selection. This operation is more intricate than the relational selection because the structure of format instances is richer. For the sake of clarity, we will present this operation in two steps. In the first step we define a version of the selection that is a simple extension of the relational selection. In the second step, this definition is generalized in order to take advantage of the semantic connections between attributes, that is, the implicit joins specified by a format.

We first extend the notion of condition on attributes. In the relational model, a condition on an attribute A is an algebraic expression in which constants (i.e. elements of the domain of A) explicitly occur. Let $f \equiv X(f_1)^*...(f_n)^*$ be a format, intuitively, since the elements of the "domain" of $f_i$ are format instances, it seems natural to consider format instances over $f_i$ as constants associated with $f_i$ in order to build conditions on $f_i$. However, writing a condition on $f_i$ according to that form is cumbersome and difficult to read. We therefore choose to implicitly specify constants associated with $f_i$ in terms of selection on $f_i$.

We now give the syntax of an expression of Verso simple selection. In the following definition, the notion of a condition on a set of attributes is the one given in the Preliminaries.

**Definition:** Let $f \equiv X(f_1)^*...(f_n)^*$. Then an expression (or operator) $S$ of Verso simple selection over f is of the form $S \equiv X{:}C\ (e_1(S_1)...e_n(S_n))$ where:

1. C is a condition over the set of attributes X,
2. $S_i$ is an expression of Verso simple selection over $f_i$, for i=1..n and,
3. $e_i \in \{\exists, \nexists, ?\}$, for i=1..n. ($\exists$ is read *"exists"*, $\nexists$ *"does not exist"* and ? *"does not care"*)

**Example 3.2:** Let $f \equiv COURSE(STUDENT(GRADE)^*)^*$ be a format. Consider the two following queries:

1. $Q_1$: "Give the list of math STUDENTs who got a GRADE greater than 10 and the GRADEs grater than 10 these STUDENTs received."
2. $Q_2$: "Give the COURSEs in which at least a STUDENT is registered that didn't get any GRADE and list the name of the STUDENTs who have no GRADE."

The query $Q_1$ is expressed by the following expression of Verso simple selection over f.

$S_1 \equiv$ COURSE : COURSE = "math"

(?( STUDENT : ( ∃ (GRADE : GRADE ≥ 10 ))))

The query $Q_2$ is expressed by the following expression of Verso simple selection over f.

$S_2 \equiv$ COURSE : ( ∃ ( STUDENT : ( ∄ ( GRADE ))))

□

An expression of Verso simple selection $S$ over f defines an operation on Inst(f) in the following way:

**Definition:** Let $S \equiv X{:}C\ (e_1(S_1)...e_n(S_n))$ be an expression of Verso simple selection over $f \equiv X(f_1)^*...(f_n)^*$. Let $e \in \{\exists, \nexists, ?\}$ then I <u>satifies</u> e, denoted $I \models e$, iff $e \equiv ?$ or $e \equiv \exists$ and $I \neq \emptyset$, or $e \equiv \nexists$ and $I = \emptyset$.

Then the <u>result of</u> $S$ <u>applied to</u> I, denoted $S(I)$, is defined by:

$S(I) = \{ < u S_1(I_1)...S_n(I_n) > | < u I_1...I_n > \in I$ such that $u \models C$, and $S_i(I_i) \models e_i$, for $i = 1..n\}$.

The previous definition is illustrated in Figure 3-5 using the Verso simple selection expression $S_1$ and $S_2$ of Example3.2.

We now propose an extension of the preceding definitions which dramatically increases the power of the operation. Consider the following query $Q_3$ on COURSE(STUDENT(GRADE)$^*$)$^*$:"Give the list of COURSEs, STUDENTs and

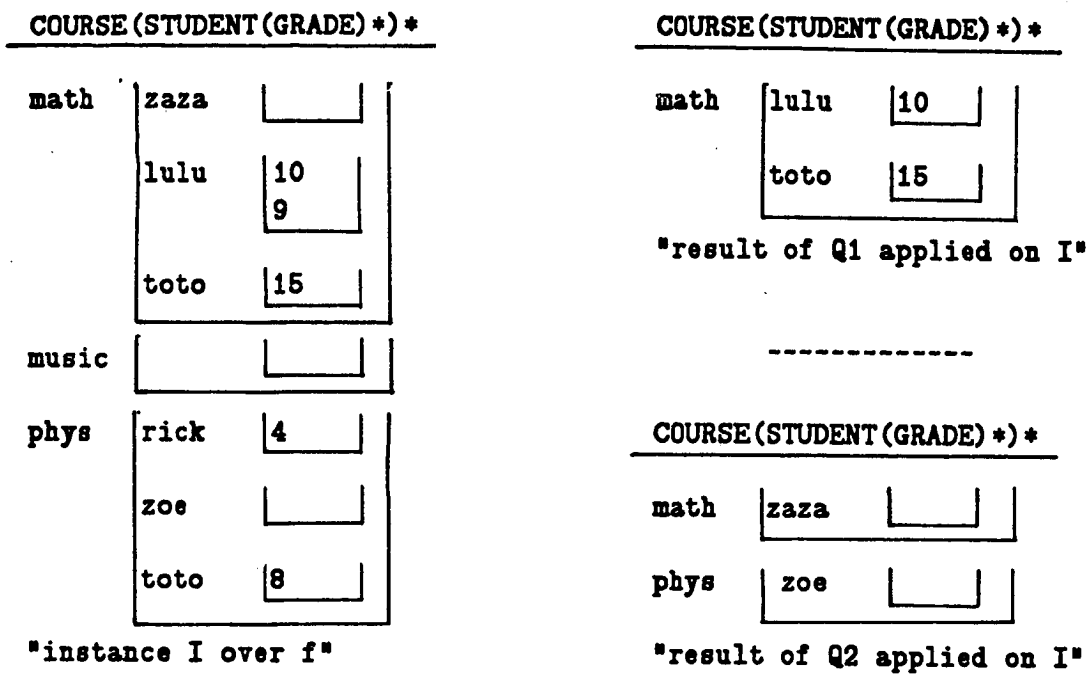


**Figure 3-5:** Verso simple selection.

GRADEs such that the STUDENT "toto" got a GRADE equal to "15" in the COURSE and a STUDENT (not necessarily "toto") got a GRADE equal to "10" in the course". Note that this query is complicated by the fact that they are several roles for the same attribute, namely STUDENT. Such a query would typically require several joins in the conventional relational model. What we mean by such a query is, in fact, two selections on Grade, say $S_1 \equiv$GRADE:GRADE="15" and $S_2 \equiv$GRADE:GRADE="10". Now, we also need two selections on STUDENT(GRADE)*:

$S_1' \equiv$ STUDENT : STUDENT = "toto" ( $\exists$ ($S_1$) ).

$S_2' \equiv$ STUDENT : ( $\exists$ ($S_2$) ).

The first one filters the STUDENT "toto" if he got a GRADE equal to "15", and the second one filters any STUDENT who got a GRADE equal to "10". We can express the complete query by:

$S \equiv$ COURSE : ( ? ($S'$) | { $\exists(S_1')$ , $\exists(S_2')$ } ) , where $S'$ is the identity on STUDENT(GRADE)$^*$.

It should be noted that the expression $S$ is not an expression of Verso simple selection as defined above. Intuitively, when we perform such a selection on an instance I over COURSE(STUDENT(GRADE)$^*$)$^*$, for each element $<uI_1>$ of I, we perform $S_1'$ and $S_2'$ "in parallel" on $I_1$ and we write $<uI_1>$ ( i.e. $<uS'(I_1)>$ ) iff $S_1'(I_1)\neq\emptyset$ and $S_2'(I_1)\neq\emptyset$. Note that here, $S_1'$ and $S_2'$ are used exclusively as conditions that validate or invalidate the result of applying the selection COURSE : ( ? ($S'$) ) on $<uI_1>$.

Formally, we define an expression of Verso selection as follows:

**Definition:** Let $f\equiv X(f_1)^*...(f_n)^*$ be a format. Then an <u>expression of Verso selection</u> over f is recursively defined as follows:

1. An expression of Verso simple selection over f is an expression of Verso selection over f, and
2. For i=1..n, let $\hat{S}_i$ be a set of expressions of the form $e'(S')$ where $e'\in\{\exists,\nexists\}$ and $S'$ is an expression of Verso selection over $f_i$. Then:

   X:C $(e_1(S_1)|\hat{S}_1,...,e_n(S_n)|\hat{S}_1)$ is an expression of Verso selection over f, where C is a condition over X, $S_i$ is a Verso selection over $f_i$ and $e_i\in\{\exists,\nexists,?\}$.

The result of a Verso selection operator over a format f applied to an instance over f is defined below:

**Definition:** Let $X:C\ (e_1(S_1)|\hat{S}_1,...,e_n(S_n)|\hat{S}_1)$ be a Verso selection over the format $f\equiv X(f_1)^*...(f_n)^*$. Let I be an instance over f. Then the result of $S$ applied to I is the instance over f,denoted by $S(I)$, defined by:

$$S(I)=\{<uS_1(I_1)...S_n(I_n)>|<uI_1...I_n>\in I \text{ such that } u\models C,$$
$$\text{for } i=1..n\ S_i(I_i)\models e_i \text{ and } S'(I_i)\models e' \text{ for each } e'(S')\in\hat{S}_1\}$$

In Figure 3-6, the result of the Verso selection $S$ applied to the instance I of Figure 3-5 is presented.

COURSE (STUDENT (GRADE) *) *

| | | |
|---|---|---|
| math | zaza | |
| | lulu | 10<br>9 |
| | toto | 15 |

"result of Q3 applied on I"

**Figure 3-6:** Verso selection.

As stated by Theorem3.6, the operation of Verso selection can be decomposed into Verso simple selections involving only conditions of the form "$\exists$" (called $\exists$-V-selection), Verso projections, unions, differences and joins. Two preliminary Lemmas are needed for this result.

We first decompose the Verso simple selection operation in a restricted case. A format f is said linear if the tree associated with f has only one branch.

**Lemma 3.3:** Let f be a linear format and $S$ an expression of Verso simple selection over f. Then, $S$ is equivalent[2] to a Verso expression of ∃-V-selection, Verso projection and difference.

**Proof:** Let f be a linear format and $S$ be an expression of Verso simple selection over f then, in order to show that ($\alpha$) $S$ is equivalent to a Verso expression of ∃-V-selection, Verso projection and difference, we proceed by an induction on $|\#(f)-\#(S)|$ where

i. if $f \equiv X$ then $\#(f)=0$ and if $f \equiv X(f_1)^*$, $f_1$ non empty, $\#(f)=1+\#(f_1)$.

ii. if $S$ is an expression of Verso simple selection over X then $\#(S)=0$, if $S \equiv X:C(e(S_1)$ is an expression of over $X(f_1)^*$, then if $e=\exists$, $\#(S)=1+\#(S_1)$ else $\#(S)=0$.

.....

1. $|\#(f)-\#(S)|=0$, then $\#(f)=\#(S)$ and by definition of $\#(S)$, $S$ is an expression of ∃-V-selection. Thus ($\alpha$) is satisfied.

2. Assume now that for each linear format f and each expression of Verso simple selection $S$ with $|\#(f)-\#(S)|<p$, ($\alpha$) is satisfied. Let $f \equiv X(Y_1(...(Y_n)...)^*$, $Y_1,...,Y_n$ non empty, be a linear format and $S$ an expression of Verso simple selection over f such that $|\#(f)-\#(S)|=|n-\#(S)|=p$. Let $l=\#(S)$, then $S$ is of the form $X:C(e_1(Y_1:C_1(...(e_n(Y_n:C_n)...)$ where for $i=1..l$, $e_i=\exists$ and $e_{l+1} \in \{?, \not\exists\}$.

   Consider the Verso simple selection over f defined by $S' \equiv X:C(e'_1(Y_1:C_1(...(e'_n(Y_n:C_n)...)$ where for $i=1..l+1$, $e'_i=\exists$ and for $i=l+2..n$, $e'_i=e_i$. As $|n-(l+1)|<p$, by induction hypothesis ($\alpha$) is satisfied for f and $S'$.

   Consider the Verso simple selction over $g \equiv X(Y_1(...(Y_l)...)^*$ $S_a \equiv X:C(e_1(Y_1:C_1(...(e_l(Y_n:C_n)...)$ . As $S_a$ is a ∃-V-selection over g, ($\alpha$) is satisfied for g and $S_a$.

   a. If $e_{l+1}=?$, then $S$ is equivalent to $(S_a \circ [g]) \oplus_f (S')$ by definition of the Verso union, selection and projection.

---

[2]The notion of equivalence between Verso expression is defined in the standard way.

b. If $e_{l+1}=\not\exists$, then $S$ is equivalent to $(S_a \circ [g]) \ominus_f ([g] \circ S')$ by definition of the Verso difference, selection and projection.

Thus, in both cases, ($\alpha$) is satisfied.

□

This result can be extented to arbitrary formats using the following result:

**Lemma 3.4:** Let $S \equiv X{:}C\ (e_1(S_1)...e_n(S_n))$ be an expression of Verso simple selection over $f \equiv X(f_1)^*...(f_n)^*$ where for i=1..n, $f_i$ is a format on $Y_i$. Let $j \in [1..n]$, $P'=X \cup Y_j$ and $P''=X \cup_{i=1..n, i \neq j} Y_i$. Then $S$ is equivalent to the Verso expression $(S' \circ [f_{|P'}]) \circledast_f (S'' \circ [f_{|P''}])$ where:

1. $S' \equiv X{:}C\ (e_j(S_j))$ is an expression of Verso simple selection over $f_{|P'}$, and
2. $S'' \equiv X{:}C\ (e_1(S_1)...e_{j-1}(S_{j-1})e_{j+1}(S_{j+1})...e_n(S_n))$ is an expression of Verso simple selection over $f_{|P''}$

The proof of this result is straightforward from the definitions of the Verso projection and selection. Intuitively, this result allows us to decompose an expression of Verso simple selection over a format $X(f_1)^*...(f_n)^*$ on the linear subformats $X(f_i)^*$, i=1..n. Then from Lemma 3.3 and 3.4, we deduce that Verso simple selections over f can be expressed by a Verso query formed of ∃-V-selections, projections, unions, differences and joins.

The Verso selection operation is also decomposed in a restricted case:

**Lemma 3.5:** Let $X{:}C\ (e_1(S_1)|\{e_1'(S_1')\} \cup \tilde{S}_1)$ be an expression of Verso selection over the format $f \equiv X(f_1)^*$. Let $S_a \equiv X{:}C\ (e_1(S_1)|\tilde{S}_1)$ and $S_b \equiv X{:}C\ (e_1'(S_1')|\emptyset)$ be two expressions of Verso selection over f. Then, $S$ is equivalent to $S_a \circledast_f ([X] \circ S_b)$

Note that by recursively applying the above decomposition, a Verso selection can be expressed exclusively in terms of Verso simple selections, joins and projections. Thus, from Lemmas 3.3, 3.4 and 3.5 we have:

**Theorem 3.6:** Let $S$ be an expression of Verso selection over the format f, then $S$ is equivalent to a Verso expression involving only the following operations: Verso projection, ∃-V-selection, union, difference and join.

The second part of this section is devoted to some results concerning the connection between the Verso model and the relational model. Only those results that are needed in the discussion of section are presented.

We first associate a format with a relational database schema called format skeleton, that intuitively give a non hierarchical description of the data structure induced by a format.

**Definition:** Let f be a format. Then the format skeleton of f, denoted Skel(f), is the relational database schema recursively defined by:

1. if $f \equiv X$ then $Skel(f)=\{X\}$, and
2. if $f \equiv X(f_1)^*...(f_n)^*$, then $Skel(f)=\{X\} \cup \{XY \mid Y \in Skel(f_i)$, for some $i \in [1..n]\}$.

For example the database schema associated with the format $COURSE(BOOK)^*(STUDENT(GRADE)^*)^*$ is R={ {COURSE}, {COURSE,BOOK}, {COURSE,STUDENT}, {COURSE,STUDENT,GRADE} }.

Using the notion of format skeleton, the information contents of format instances can be "described" by relational database instances.

**Definition:** Let f be a format and I an instance over f. The instance skeleton of I, denoted skel(I), is the relational database instance over Skel(f) defined by:

1. if $f \equiv X$ then $skel(I)(X)=I$, and
2. if $f \equiv X(f_1)^*...(f_n)^*$, then $skel(I)(X)=\{u \mid <uI_1...I_n> \in I\}$ and, $skel(I)(XY)=\cup_{<uI_1...I_n> \in I}\{uXskel(I_i)(Y)\}$, $Y \in Skel(f_i)$ for some $i \in [1..n]$.

It is clear that not all database schemas are associated with some formats and furthermore not all database instances defined over a format skeleton are associated with format instances. We characterize below the database instances over format skeleton that are instance skeletons.

**Theorem 3.7:** [Abiteboul + Bidoit 84b] Let f be a format and R=Skel(f). Let r be an instance over R. Then the following two assertions are equivalent:

1. r=skel(I) for some instance I over f, and
2. r satisfies the URSA.

As we just defined a mapping from instances over f onto database instances over Skel(f) satisfying the URSA, we are able now to characterize the Verso operations on V-relations in terms of relational operations on relational database instances. The first result give a simple interpretation of the binary Verso operations, namely union, difference and join.

**Proposition 3.8:** [Abiteboul + Bidoit 84b] Let f and g be two compatible formats, f and g subformat of h. Let I and J be instances over f and g respectively. Let r=skel($I^h$) and s=skel($J^h$). Then:

1. skel($I \oplus_h J$)=r $\cup$ s,
2. skel($I \ominus_h J$) is the smallest[3] URSA-instance over Skel(h) containing r-s , and
3. skel($I \circledast_h J$) is the largest URSA-instance over Skel(h) contained in the instance t defined over Skel(h) by:
   a. t(X)=r(X)$\cap$s(X) if X$\in$Skel(f)$\cap$Skel(g),
   b. t(X)=r(X) if X$\in$Skel(f)-Skel(g),
   c. t(X)=s(X) if X$\in$Skel(g)-Skel(f), and
   d. t(X)=$\emptyset$ otherwise.

We now characterize the two unary Verso operations of projection and selection.

**Theorem 3.9:** [Bidoit 84] Let f be a format over Z, P$\subseteq$Z such that $f_{|P}$ is a format and let g be $f_{|P}$. Let I be an instance over f, r=skel(I) the instance skeleton of I over Skel(f). Then skel(I[g]) is the smallest URSA-instance over Skel(g) and containing $\Pi_P$(r).

Since it has been shown earlier that the Verso selection operation can be decomposed using the Verso operations of projection, $\exists$-V-selection, union, difference and join , we only present here the interpretation of $\exists$-V-selection.

**Theorem 3.10:** [Bidoit 84] Let f be a format and *S* be an expression of $\exists$-V-selection over f. Let I be an instance over f and r=skel(I). Then:

---

[3] Let r and s be two database instances over R, r$\leq$s iff $\forall$X$\in$R r(X)$\subseteq$s(X).

$\forall Z \in Skel(f)$, $skel(S(I))(Z) = \Pi_Z(\sigma)$ where $\sigma$ is the relational instance defined over $\cup_{Z \in Skel(f)} Z$ by:

$\sigma = \bowtie_{Z \in Skel(f)} [Select_{\wedge_{Y \subseteq Z} C_Y} r(Z)]$ with $C_Y$ is a condition on Y occuring in $S$.

## 4. Schema tableau to represent PSJ-expressions

Tableaux have been successfully used to optimize a subset of relational expressions and for testing implication of data dependencies [Maier + Mendelzon + Sagiv 79], [Aho + Sagiv + Ullman 79a], [Aho + Sagiv + Ullman 79b], [Klug 80]. In the last section, we shall show that tableaux can also be used to recognize relational expressions of projection, selection and join that can be "equilalently expressed" by a Verso selection followed by a Verso projection.

In this section, we slightly extend the standard definition of tableau:

1. In most of the previous studies only conditions of the form A=a (where A is an attribute and a in DOM(A)) are handled. We consider here conditions of more general nature over attributes.
2. Since our goal is to use tableaux to represent PSJ-expressions expressed over specific type of database schema (that is, over format skeletons), the definition of tableau has to take into account the structure of the PSJ-expressions considered. We also have to slightly adjust the conventional definition of the mapping associated with a tableau since the instances on which this mapping is applied must satisfy the URSA.

In order to define a schema tableau, we need some intermediate concepts which are now presented. In a natural way, we associate with each attribute A of U an infinite set of symbols called A-variables. These sets of variables, denoted by VAR(A) for each A, satisfy the following conventional properties: for each A,B $\in$U, $VAR(A) \cap DOM(B) = \emptyset$

and $VAR(A) \cap VAR(B) = \emptyset$ if $A \neq B$. The definition of a condition over an attribute A is the same as the definition of a condition over the singleton {A} given in the Preliminaries.

We define a schema tableau as a set of particular tuples called criteria.

**Definition:** Let A be an attribute and V a set of attributes. Then

1. An elementary criterion on A is a pair $\langle x_A, C \rangle$ where $x_A \in VAR(A)$ and $C$ is a condition on A. CRITER(A) denotes the set of all criteria on A. A criterion v over V is a mapping from V to $\cup_{A \in V} CRITER(A)$ such that $v(A) \in CRITER(A)$ for each A in V. CRITER(V) denotes the set of all criteria over V.

In the following, we denote by c.var (resp. c.cond) the first (resp. second) component of an elementary criterion c on A. We use, in a natural way, two operations on criteria over sets of attributes, namely projection and join. Let v ,w be criteria over V and W respectively. First assume that $W \subseteq V$ , then the projection of v over W, denoted $\Pi_W(v)$, is the criterion over W defined by : $\forall A \in W$, $\Pi_W(v)(A) = v(A)$. Assume now that $V \cap W \neq \emptyset$, then if $\forall A \in V \cap W$, v(A).var=w(A).var then the join of v and w, denoted $v \bowtie w$, is defined as a criterion over $V \cup W$ by:

1. $\forall A \in V\text{-}W$, $(v \bowtie w)(A) = v(A)$, and
   $\forall A \in W\text{-}V$, $(v \bowtie w)(A) = w(A)$
2. $\forall A \in V \cap W$, $(v \bowtie w)(A).var = v(A).var = w(A).var$ and,
   $\forall A \in V \cap W$, $(v \bowtie w)(A).cond = v(A).cond \wedge w(A).cond$;

The formal definition of a schema tableau follows Figure 4-1 that presents a schema

tableau over R={ {COURSE}, {COURSE,BOOK}, {COURSE,STUDENT} } on the form of a table where each column is associated with an attribut and each row represents a criterion. For sake of space, the conditions of the elementary criteria in the table do not contain attribute names that are implicit by the columns they are contained in.

| COURSE | STUDENT | BOOK | |
|---|---|---|---|
| (x1, ="math") | (y2° True ) | | summary |
| (x1, ="math") | (y2, True ) | | T |
| (x1, ="math") | (y2, ="toto") | (z1, ="b1") | |

**Figure 4-1:** A Schema Tableau over R.

The formal definition of a schema tableau is given below:

**Definition:** Let R be a database schema. Then a schema tableau **T** over R is a pair $(t_0,T)$ where

1. T is a finite set of criteria such that:
   a. $\forall v \in T$ , $v \in CRITER(V)$ implies $\exists S \subseteq R \mid V = \cup_{Y \in S} Y$.
   b. $\forall v,w \in T$, $v(A).var = w(A).var \Rightarrow v(A).cond = w(A).cond$
2. each elementary criterion of $t_0$ appears in T.

The set of elementary criteria occuring in **T** is denoted CRITER(**T**).

Following the standard procedure, we now associate a mapping on the set of database instances over R with a schema tableau **T** over R. We consider only database instances over R satisfying the URSA. First, the notion of a valuation of a set of variable is introduced.

**Definition:** Let H be a subset of $\cup_{A\in U}VAR(A)$. Then a valuation $\rho$ of H is a mapping from H to $\cup_{A\in U}DOM(A)$ such that $x\in VAR(A)$ entails $\rho(x)\in DOM(A)$.

Let H be a set of elementary criteria. Then a valuation $\rho$ of H is a mapping from $\{x|\exists c\in H, x=c.var\}$ to $\cup_{A\in U}DOM(A)$ such that: $\forall c\in H$, $\rho(c.var)\models c.cond$.

In the following, if $\rho$ is a valuation of a set H of elementary criteria and c is in H, $\rho(c)$ designates $\rho(c.var)$. As a straighforward generalization of the preceding definition, a valuation $\rho$ of a set of criteria T is defined as a valuation $\rho$ of the set CRITER(T) of elementary criteria occuring in T. If v is in T and v is a criterion over V, $\rho(v)$ denotes the tuple defined over V by $\rho(v)(A)=\rho(v(A))$ for each A in V and $\rho(T)$ denotes the set of tuples $\rho(v)$ such that v is in T.

We now exhibit the mapping on relational database instances over a schema R and satisfying the URSA, induced by a given schema tableau over R. First, a containment relationship between a valuation of a schema tableau **T** over R and an URSA-instance r over R is given by the definition below:

**Definition:** Let R be a database schema and $\mathbf{T}=<t_0,T>$ be a schema tableau over R. Let $\rho$ be a valuation of **T** and r be an instance over R satisfying the URSA. Then $\rho$ is a valuation of T into r iff the instance s defined over R by $\forall V\in R$ $s(V)=\Pi_V(\rho(T))$ is included in r.

**Definition:** Let R be a database schema and $\mathbf{T}=<t_0,T>$ be a schema tableau over

R. Let $V_0$ be the set of attributes such that $t_0 \in CRITER(V_0)$. Then, $V_0$ is called the target schema of **T** and **T** defines the mapping from the set of URSA-instances over R into $REL(V_0)$ as follows:

For each URSA-instance r over R, $T(r)=\{\rho(t_0) | \rho$ is a valuation of T into r$\}$

In Figure 4-2, an instance r over R={ {COURSE}, {COURSE,STUDENT}, {COURSE,BOOK} } is represented as well as the instance T(r) defined over {COURSE,STUDENT} where **T** is the schema tableau given in Figure 4-1.

| COURSE |
|---|
| math |
| phys |
| cs |

| COURSE | STUDENT |
|---|---|
| math | toto |
| math | lulu |
| phys | zoe |

| COURSE | BOOK |
|---|---|
| math | b1 |
| phys | b2 |
| cs | b1 |

"Instance r over R satisfying the URSA"

| COURSE | STUDENT |
|---|---|
| math | toto |
| math | lulu |

"result of the mapping associated with the schema tableau T applied to the instance r"

**Figure 4-2:** Mapping associated with a schema tableau.

Before showing how to build a tableau defining the same mapping as a PSJ-expression, we give some results characterizing equivalent tableaux. Intuitively, two schema tableaux on R are equivalent if they induce the same mapping on the URSA-instances over R. Formally:

**Definition:** Let R be a database schema. Let **T** and **T'** be two schema tableau over R and having same target schema. Then **T** is <u>included in</u> **T'**, denoted $\mathbf{T} \subseteq \mathbf{T}'$,iff $\mathbf{T}(r) \subseteq \mathbf{T}'(r)$ for each URSA-instance r over R. And **T** and **T'** are <u>equivalent</u>, denoted $\mathbf{T} \equiv \mathbf{T}'$, iff $\mathbf{T} \subseteq \mathbf{T}'$ and $\mathbf{T} \supseteq \mathbf{T}'$.

We first show that two tableaux are equivalent if they are "equal" modulo some renaming of variables. To present this first result, we use the notion of a schema tableau version.

**Definition:** Let c and c' be two criteria on A. Then c' is a version of c iff $c.cond \equiv c'.cond$.

Let $\mathbf{T}=(t_0,T)$, $\mathbf{T}'=(t_0',T')$ be two schema tableaux over R. Then **T** is a <u>version of</u> **T'** iff there exists a 1-1, onto mapping $\nu$ from CRITER(**T**) into CRITER(**T'**) such that:

1. $\forall\, c \in \text{CRITER}(\mathbf{T})$, $\nu(c)$ is a version of c,
2. $\nu(t_0)=t_0'$, and
3. $\nu(T)=\{\nu(v) | v \in T\}=T'$.

Now we have the following result:

**Theorem 4.1:** Let **T** and **T'** be two schema tableaux over R. If **T** is a version of **T'** then **T** and **T'** are equivalent.

The second result gives a characterization of schema tableau inclusion. In order to

present this result, we need to define a mapping on schema tableau called containment mapping.

**Definition:** Let $\mathbf{T}=(t_0,T)$, $\mathbf{T'}=(t_0',T')$ be two schema tableaux over R. Let $\theta$ be a mapping from CRITER(**T**) into CRITER(**T'**). Then $\theta$ is a containment mapping from **T** into> **T'** iff:

1. $\theta(t_0)=t_0'$, and
2. $\forall$ v$\in$T$\cap$CRITER(V) and $\forall$Z$\in$R , Z$\subseteq$V implies: $\exists$ v'$\in$T'$\cap$CRITER(V') such that:
   a. Z$\subseteq$V',
   b. $\forall$ A$\in$Z, v'(A).cond $\Rightarrow$ v(A).cond, and
   c. $\Pi_Z[\theta(v)]=\Pi_Z[v']$.

We have then:

**Theorem 4.2:** Let **T**, **T'** be two schema tableaux over R. Then **T**$\subseteq$**T'** iff there exist a containment mapping $\theta$ from **T'** into **T**.

Before giving the proof of Theorem 4.2, two examples are presented here to intuitively justify respectively conditions 2.b and 2.c of the containment mapping definition.

**Example 4.3:** Let R={{COURSE},{COURSE,STUDENT},{COURSE,BOOK}} be a relational database schema and consider the two tableaux $\mathbf{T_1}=(t_1,T_1)$ and $\mathbf{T_2}=(t_2,T_2)$ over R represented in Figure 4-3. Let $\theta_1$ be the identity mapping from CRITER($T_2$) into CRITER($T_1$).

It is easy to check that $\theta_1$ satisfies the definition of a containment mapping from $\mathbf{T_2}$ into $\mathbf{T_1}$. Note that $\theta_1$ maps both the first and second "rows" of $T_2$ on the single "row" of $T_1$.

Note also here that $\theta_1$ is a bijection and $\theta_1^{-1}$ is a containment from $\mathbf{T_1}$ into $\mathbf{T_2}$. Then by Theorem 4.2 and the definition of tableau equivalence, we have: $\mathbf{T_1} \equiv \mathbf{T_2}$. In fact, we will see later that $\mathbf{T_2}$ is the schema tableau over R associated with the relational expression

$$\Pi_{COURSE}[\text{select}_{COURSE=\text{"math"}}[[COURSE,STUDENT] \bowtie [\text{select}_{BOOK=\text{"b1"}}[COURSE,BOOK]]]]$$

and $\mathbf{T_1}$ is obtained from $\mathbf{T_2}$ using the join operator on criteria. □

**Example 4.4:** Let R be the relational database schema of Example 4.3. Consider the tableaux $\mathbf{T_2}=(t_2,T_2)$ and $\mathbf{T_3}=(t_3,T_3)$ represented in Figure 4-3. Let $\theta_2$ be the mapping from CRITER($T_3$) into CRITER($T_2$) defined by:

$$\theta_2((x1,math))=(x1,math),$$

$$\theta_2((z1,=\text{"b1"} \vee =\text{"b2"}))=(z1,=\text{"b1"}).$$

Note that $\theta_2$ maps the first (respectively second) "row" of $T_3$ on either "row" of $T_2$ (respectively on the second "row" of $T_2$) and that $\theta_2((z1,=\text{"b1"} \vee =\text{"b2"}) \Rightarrow (z1,=\text{"b1"} \vee =\text{"b2"})$. Then $\theta_2$ is a containment mapping from $\mathbf{T_3}$ into $\mathbf{T_2}$ and by Theorem 4.2 we have: $\mathbf{T_2} \subseteq \mathbf{T_3}$. In fact, we will see later that $\mathbf{T_3}$ is the schema tableau over R associated with the relational expression

$$\Pi_{COURSE}[[\text{select}_{COURSE=\text{"math"}}[COURSE] \bowtie [\text{select}_{BOOK=\text{"b1"} \vee BOOK=\text{"b2"}}[COURSE,BOOK]].$$

Note here that there is no containment mapping from $\mathbf{T}_2$ into $\mathbf{T}_3$. Thus $\mathbf{T}_2$ and $\mathbf{T}_3$ are not equivalent. Neither are the relational expression given in Example 4.3 and the relational expression given above.

□

$\mathbf{T}_1$

| COURSE | STUDENT | BOOK |
|---|---|---|
| (x1,="math") | | |
| (x1,="math") | (y1,True) | (z1,="b1") |

$\mathbf{T}_2$

| COURSE | STUDENT | BOOK |
|---|---|---|
| (x1,="math") | | |
| (x1,="math") | (y1,True) | |
| (x1,="math") | | (z1,="b1") |

$\mathbf{T}_3$

| COURSE | STUDENT | BOOK |
|---|---|---|
| (x1,="math") | | |
| (x1,="math") | | |
| (x1,="math") | | (z1,="b1" ="b2") |

**Figure 4-3:** Containment mapping and tableau equivalence.

We now proceed to the proof of Theorem 4.2.

**Proof:**

1. Assume first that $\mathbf{T}=(t_0,T)$ and $\mathbf{T}'=(t_0',T')$ are two schema tableaux over R such that ($\beta$) there exists a containment mapping $\theta$ from $\mathbf{T}'$ into $\mathbf{T}$. We show that $\mathbf{T}(r)\subseteq\mathbf{T}'(r)$ for each URSA-instance r over R. Let r be an URSA-instance over R and let $\rho$ be a valuation of T into r. Consider then $\rho'$ defined over CRITER(T') by: $\rho'(v')=\rho(\alpha(v'))$ for each $v'\in$CRITER(T'). We show that $\rho'$ is a valuation of T' into r:

a. Let $v=\theta(v')$, then by definition of a containment mapping we have v.cond$\Rightarrow$v'.cond. To establish that $\rho'$ is a valuation of T', it is sufficient to use the fact that $\rho$ is a valuation of T.

b. Consider now s and s' respectively defined by:

i. $\forall Z\in R$, $s(Z)=\Pi_Z(\rho(T))$,

ii. $\forall Z\in R$, $s'(Z)=\Pi_Z(\rho'(T'))$.

Let W be a criterion in T defined over W and let Z be in R such that $Z\subseteq W$. Using the definition of a containment mapping, we have:

$\exists v\in T\cap CRITER(V)$ such that $Z\subseteq V$ and $\Pi_Z(\theta(w))=\Pi_z(v)$, (i.e.) $\Pi_Z(\rho(\theta(w)))=\Pi_Z(\rho(w)=\Pi_z(\rho(v))$.

From this, we deduce that $(\alpha_1)$ $s'(Z)\subseteq s(Z)$ for each Z in R. By hypothesis, $\rho$ is a valuation of T into r, then $(\alpha_2)$ $s(Z)\subseteq r(Z)$ for each Z in R. From $(\alpha_1)$ and $(\alpha_2)$, we have $s'(Z)\subseteq s(Z)\subseteq r(Z)$ for each Z in R, and then $\rho'$ is a valuation of T' into r.

We showed below that for each valuation $\rho$ of T into r there exists a valuation $\rho'$ of T' into r and as $\theta(t_0')=t_0$, $\rho(t_0)=\rho'(t_0')$, this implies that $\mathbf{T}(r)\subseteq\mathbf{T}'(r)$ as desired.

2. Assume now that $\mathbf{T}=(t_0,T)$ and $\mathbf{T}'=(t_0',T')$ are two tableau schema over R such that $\mathbf{T}\subseteq\mathbf{T}'$.

It is clear that $\mathbf{T}$ and $\mathbf{T}'$ have same target schema. It remains to exhibit a containment mapping from $\mathbf{T}'$ into $\mathbf{T}$. Let $\beta$ be a 1-1, onto mapping from CRITER(T) into a set K of constants such that $\beta(v)\in DOM(A)$ and $\beta(v)\models$v.cond if $v\in CRITER(A)$. Let r be the URSA-instance defined over R by: $\forall Z\in R$, $r(Z)=\{\Pi_Z(\beta(v))|v\in T\}$. Then $\beta(t_0)\in\mathbf{T}(r)$. By hypothesis, $\mathbf{T}\subseteq\mathbf{T}'$ then $\beta(t_0)\in\mathbf{T}'(r)$. By definition of $\mathbf{T}'(r)$, we deduce that there exists a valuation $\rho'$ of T' into r such that $\rho'(t_0')=\beta(t_0)$. Thus we have by definition of a valuation of T' into r that:

$(\alpha)$ $\{\Pi_Z(\rho'(v'))|v'\in T'\}\subseteq\{\Pi_Z(\beta(v))|v\in T\}$ for each Z in R.

Now consider the mapping $\theta=\beta^{-1}\circ\rho'$ from CRITER(T') into CRITER(T). We show that $\theta$ is a containment mapping from $\mathbf{T}'$ into $\mathbf{T}$. Let $v'\in T'\cap CRITER(V')$ and $Z\in R$, $Z\subseteq V'$.

From $(\alpha)$, there exists $v\in T\cap CRITER(V)$ such that $(\delta)$ $\Pi_Z(\rho'(v'))=\Pi_z(\beta(v))$ thus such that:

a. $Z \subseteq V$,

b. Suppose there exists $A \in Z$ such that v(A).cond does not implies v'(A).cond. Then there exists $a \in DOM(A)$ such that $a \models v(A).cond$ and $a \not\models v'(A).cond$. Let $a = \beta(v(A)) \in$. By ($\delta$), $\rho'(v'(A)) = \beta(v(A)) = a$. Hence $\rho'$ is a valuation of T', then $a \models v'(A).cond$, a contradiction. We proved here that $\forall A \in$, $v(A).cond \Rightarrow v'(A).cond$.

c. From ($\delta$) we deduce that: $\Pi_Z(\theta(v')) = \Pi((\beta^{-1} \circ \rho')(v')) = \Pi_Z((\beta^{-1} \circ \beta)(v'))$, that is $\Pi_Z(\theta(v')) = \Pi_Z(v)$.

We proved above that $\theta$ is a cntainment mapping from **T'** into **T**.

□

We now exhibit three transformations on schema tableaux that preserve the mapping they initially define (i.e. that produce equivalent tableaux). The first transformation uses the projection operation on criteria.

**Definition:** Let $\mathbf{T} = (t_0, T)$ be a schema tableau over R. Then the closure under projection of **T** according to R, denoted $\Pi(\mathbf{T})$, is the schema tableau $(t_0, \Pi(T))$ over R where $\Pi(T) = \{\Pi_Z(v) | v \in T \text{ and } Z \in R\}$.

Note that the closure under projection applied to a given schema tableau **T** over R produces a tableau with criteria over single elements of R rather than over unions of elements of R.

The second transformation is defined in terms of the join operation on criteria.

**Definition:** Let $T=(t_0,T)$ be a schema tableau over R. Then the closure under join of T, denoted $*(T)$, is the schema tableau $(t_0,*(T))$ over R defined by $*(T)=\cup_{i=1..\infty}T^i$ where $T^0=T$ and $T^{i+1}=\{v\bowtie w | v\in T, w\in T^i$ and v,w are joinable$\}$.

The effect of the closure under join is to produce a schema tableau R with as many criteria defined over unions of collections of elements of R as possible.

The last transformation on schema tableau is defined using a partial order over criteria. This partial order is defined as follows:

**Definition:** Let V,W be two sets of attributes, and v,w be two criteria defined over V, respectively W. Then we say that v is less restrictive than w, denoted $v\leq w$, iff: $V\subseteq W$, and $\forall A\in V$, v(A).var=w(A).var and w(A).cond$\Rightarrow$v(A).cond.

Now let T,T' be two sets of criteria such that $T'\subseteq T$, then T' maximaly represents$>$ T iff:

1. $\forall v\in T, \exists w\in T' \mid v\leq w$, and
2. $\forall v,w \in T', v\leq w \Rightarrow v=w$.

Note here that for any set of criteria T, there exists a subset T' of T such that T' maximally represents T but T' is not unique. For example, $\{(x_A,A<\text{"a"}\wedge A>\text{"a"})\}$ and $\{(x_A,A\neq\text{"a"})\}$ both maximally represent $\{(x_A,A<\text{"a"}\wedge A>\text{"a"}),(x_A,A\neq\text{"a"})\}$. However, if T is the set of criteria of a schema tableau $T=(t_0,T)$ then we can easily show by definition of a schema tableau that there exists a unique subset of T, denoted

Max(T) such that Max(T) maximally represents T. Let Max(**T**) denote the schema tableau ($t_0$,Max(T)).

We now show that the three preceding transformations produce equivalent tableaux.

**Proposition 4.5:** Let **T** be a schema tableau over R. Then **T**, Π(**T**), *(**T**) and Max(**T**) are equivalent.

The proof of this proposition is obvious using the Theorem 4.2.

Let **T** be a schema tableau over R. Intuitively, the schema tableau Max(Π(**T**)) is the "smallest"[4] schema tableau over R equivalent to **T** such that each criterion is defined over an element of R. On the other hand, Max(*(**T**)) is the "smallest" schema tableau equivalent to **T**. We do not formally discuss these properties here but simply use the preceding transformation to construct the schema tableau associated with a PSJ-expression.

**Definition:** Let R be a schema and E be a PSJ-expression on R. Then the schema tableau ($t_0$,T) over R associated with E, denoted $\mathbf{T}_E$, is recursively defined as follows:

1. if E≡[X] where X∈R then
   a. $t_0$∈CRITER(X) and ∀A∈X,$t_0$(A).cond=True,
   b. T={$t_0$}.
2. if E≡$\Pi_Y$[$E_1$] where $E_1$ is a PSJ-expression on R with target schema $X_1$, and $Y \subseteq X_1$. Let $\mathbf{T}_{E_1}$=($t_1$,$T_1$) then:

[4] By "smallest" we mean here the schema with the minimum number of criteria (or rows).

a. $t_0=\Pi_Y(t_1)$, and

b. $T=T_1$.

3. if $E\equiv[E_1|C]$ where $E_1$ is a PSJ-expression on R with target schema $X_1$, and $C=\wedge_{A\in X_1} C_A$ where $C_A$ is a condition over {A}. Let $\mathbf{T}_{E_1}=(t_1,T_1)$ then:

   a. $t_0\in CRITER(X_1)$ $\forall A\in X_1$, $t_0(A).var=t_1(A).var$ and $t_0(A).cond=t_1(A).cond \wedge C_A$.

   b. $T=Max(T_1 \cup \{v\bowtie w | v\in T_1, w=\Pi_Y(t_0), Y\subseteq X_1\})$.

4. if $E\equiv[E_1\bowtie E_2]$ where $E_1$, $E_2$ are PSJ-expressions on R with respectively $X_1$, $X_2$ as target schemas. Let $X_1\cap X_2\neq\emptyset$. Let $\mathbf{T}_{E_1}=(t_1,T_1)$ and $\mathbf{T}_{E_2}=(t_2,T_2)$ such that:

   let $v\in T_1$, $w\in T_2$, $v(A).var=w(A).var$ iff $A\in X_1\cap X_2$, $v(A)=t_1(A)$ and $w(A)=t_2(A)$ (Note here that if it is not the case we obviously use some version of $\mathbf{T}_1$ or $\mathbf{T}_2$). Then:

   a. $t_0=t_1\bowtie t_2$, and

   b. $T=Max((T_1\cup T_2)\cup\{v\bowtie w | v\in T_1\cup T_2$ and $w=\Pi_Y(t_0)$, for some $Y\subseteq X_1\cap X_2\}$.

Before we give an example, note that constructing **T** in part 2. of the above definition consists of projecting the summary of **T** on Y, and constructing **T** in part 3. consists of updating the conditions on the criteria of $T_1$. Intuitively, constructing **T** in part 4. consists of building the "smallest" schema tableau containing $T_1\cup T_2$.

**Example 4.6:** Consider the relational database schema R={ {COURSE}, {COURSE,STUDENT}, {COURSE,BOOK} } and the following PSJ-expression on R:

$$E\equiv\Pi_{COURSE}[select_{STUDENT=\text{"toto"}}[[COURSE,STUDENT]]\bowtie[COURSE,BOOK]].$$

Let $E_1$, $E_2$, $E_3$, $E_4$ be the relational PSJ-expressions obtained by decomposing E:

1. $E_1 \equiv$[COURSE,STUDENT],
2. $E_2 \equiv \text{select}_{\text{STUDENT}=\text{"toto"}}[E_1]$,
3. $E_3 \equiv$[COURSE,BOOK], and
4. $E_4 \equiv E_2 \bowtie E_3$.

The schema tableaux over R associated with $E_1$, $E_2$, $E_3$, $E_4$ and E are represented in Figure 4-4. □

| COURSE | STUDENT | BOOK |
|---|---|---|
| (x1,T) | (y1,T) | |
| (x1,T) | (y1,T) | |

"tableau associated with E1"

| COURSE | STUDENT | BOOK |
|---|---|---|
| (x1,T) | (y1,="toto") | |
| (x1,T) | (y1,="toto") | |

"tableau associated with E2"

| COURSE | STUDENT | BOOK |
|---|---|---|
| (x1,T) | | (z1,T) |
| (x1,T) | | (z1,T) |

"tableau associated with E3"

| COURSE | STUDENT | BOOK |
|---|---|---|
| (x1,T) | (y1,="toto") | (z1,T) |
| (x1,T) | (y1,="toto") | |
| (x1,T) | | (z1,T) |

"tableau associated with E4"

| COURSE | STUDENT | BOOK |
|---|---|---|
| (x1,T) | | |
| (x1,T) | (y1,="toto") | |
| (x1,T) | | (z1,T) |

"tableau associated with E"

**Figure 4-4:** Constructing the Schema Tableau associated with E.

We now show that a PSJ-expression on R and its associated schema tableau represent the same mapping on URSA-instances defined over R.

**Theorem 4.7:** Let R be a database schema. Let E be a PSJ-expression on R. Then $T_E(r)=E(r)$ for each instance r over R satisfying the URSA.

The proof is straightforward and uses an induction on the depth of the PSJ-expression E.

## 5. Efficient evaluation of relational PSJ-expressions by Verso selection and projection

In [Abiteboul + Bidoit 84b], it is showed that the Verso algebra is "complete". By "complete", it is meant that all queries that can be expressed by the relational algebra can also be expressed by the Verso algebra. In this section, we investigate the expressive power of the Verso selection. We exhibit a very large set of relational queries which can be simulated by a Verso selection followed by a Verso projection. The characterization of this class of relational queries uses the schema tableau representation of PSJ-expressions and leads to a constructive method for recognizing such expressions.

We begin by presenting a query which would typically require joins in the relational model but can be simply expressed by a selection in the Verso model.

**Example 5.1:** Consider the format f≡COURSE(STUDENT)*(EXAM-DAY)* and the query:"What are the COURSEs taken by the STUDENT "toto" which have an EXAM

on "November first"?". In the relational model, there would typically be two relations COURSE STUDENT and COURSE EXAM-DAY, and the query would require a join operation. This query can be formulated the following Verso simple selection:

$S\equiv$ COURSE : ( $\exists$ (STUDENT : STUDENT = "toto" ),
$\exists$ (EXAM-DAY : EXAM-DAY = "November 1st" ))

Indeed, some very natural queries like "Give the list of courses with no known exam day" can be answered by a Verso simple selection whereas they would require the use of difference in the relational model. □

We restrict our study to relational expressions of selection, projection and join. We will successively characterize:

1. PSJ-expression that can be formulated in terms of a $\exists$-V-selection,
2. PSJ-expression that can be formulated in terms of a $\exists$-V-selection followed by a Verso projection,
3. PSJ-expression that can be formulated in terms of a Verso simple selection (with no $\not\exists$ symbols called a $\exists$-?-V-selection) followed by a Verso projection, and
4. PSJ-expression that can be formulated in terms of a Verso selection (with no $\not\exists$ symbols) followed by a Verso projection.

We first need to define the notion of equivalence between Verso expressions and relational expressions. Intuitively, a relational expression E on a database schema R and a Verso expression $\mathcal{E}$ on a format f such that R=Skel(f) are equivalent iff for each URSA-instance r over R, the result of applying $\mathcal{E}$ on I such that r=skel(I) is "equal"to the relation obtained by applying E on r. Formally, we have:

**Definition:** Let f be a format and R=Skel(f). Let $\mathcal{E}$ be a Verso expression on f and E be a relational expression on R. Then E and $\mathcal{E}$ are <u>equivalent</u> (or $\mathcal{E}$ <u>translates</u> E) iff :

$\forall$ r$\in$Rel(R), r satisfying the URSA, $E(r)=\bowtie_{Z\in R} skel(\mathcal{E}(I))(Z)$ , where r=skel(I).

First, we characterize the PSJ-expressions that can be translated by $\exists$-V-selection, and in order to do this we first consider PSJ-expressions E on format skeleton R such that the set of elementary sub-expressions [Z] of E covers R.

**Definition:** Let R be a relational database schema and E be a PSJ-expression on R. Let R/E denote the set {Z|Z$\in$R and [Z] is a sub-expression occuring in E}, then E is <u>total on</u> R iff $\cup_{Z\in R} Z=\cup_{Z\in R/E} Z$ .

We now state the first result:

**Lemma 5.2:** Let f be a format and R=Skel(f). Let E be a PSJ-expression total on R. Let $\mathbf{T}_E=(t_0,T)$. Then, there exists an expression of $\exists$-V-selection on f that translates E iff $Max(^*(\mathbf{T}_E))=(t_0,\{t_0\})$ and $t_0\in CRITER(\cup_{Z\in R} Z)$.

**Proof:**

1. Assume that there is an expression of $\exists$-V-selection $S$ on f such that $S$ translates E. We show that:

   ($\alpha$) $Max(^*(\mathbf{T}_E))=(t_0,\{t_0\})$ and $t_0\in CRITER(\cup_{Z\in R} Z)$, where $\mathbf{T}_E=(t_0,T)$ is the schema tableau over R associated with E.

   From Theorem 3.10 for each instance I over f we have: $\forall Z\in R$, $skel(S(I))=\Pi_Z(\sigma)$ where $\sigma=\bowtie_{Z\in R} select_{\wedge_{Y\subseteq Z} C_Y}[skel(I)(Z)]$ and $C_Y$ is a condition over the set of attributes Y occuring in $S$.

By hypothesis, $S$ translates E, so by definition we have $\bowtie_{Z\in R}$ skel($S$(I))(Z)=E(skel(I)), that is $\sigma$=E(skel(I)) for each instance I over f.

Thus, to prove ($\alpha$) we have to prove that the schema tableau $\mathbf{T}'=(t_0',T')$ associated with $E'=\bowtie_{Z\in R}\text{select}_{\wedge_{Y\in Z}C_Y}[Z]$ satisfies $\text{Max}(*(\mathbf{T}'))=(t_0',\{t_0'\})$ where $t_0'\in\text{CRITER}(\cup_{Z\in R}Z)$. Let $(v_z,\{v_z\})$ be the shema tableau over R associated with $\text{select}_{\wedge_{Y\in Z}C_Y}[Z]$ for each $Z\in R$. By definition, we have:

a. $T'=\cup_{Z\in R}\{v_Z\}$ , $v_Z\in\text{CRITER}(Z)$,

b. $\forall Z_1,Z_2\in R$ , $\forall A\in Z_1\cap Z_2$, $v_{Z_1}(A)=v_{Z_2}(A)$, and

c. $t_0'=\bowtie_{Z\in R}v_Z$.

Then $\text{Max}(*(\mathbf{T}'))=(t_0',\{t_0'\})$.

2. Conversely, assume that ($\beta$) $\text{Max}(*(\mathbf{T}))=(t_0,\{t_0\})$ and $t_0\in\text{CRITER}(\cup_{Z\in R}Z)$ where $\mathbf{T}$ is the tableau associated with E. Let us construct the expression of $\exists$-V-selection $S$ that translates E. We naturally proceed by induction on the cardinality of R=Skel(f), denoted #(R).

a. if #(R)=1 then f≡X. Consider $S\equiv$X:C where $C=\wedge_{A\in X}t_0(A).\text{cond}$. It is obvious that $S$ translates E.

b. Assume now that for each PSJ-expression E total on R such that #(R)<p and ($\beta$) is satisfied, there exists a $\exists$-V-selection over f that translates E. Now, consider R=Skel(f) such that #(R)=p. Let E be a PSJ-expression total on R satisfying ($\beta$). Let $f\equiv X(f_1)^*...(f_n)^*$. For i=1..n, let $g_i\equiv X(f_i)^*$ and consider the schema tableau $\mathbf{T}_i=(t_i,\{t_i\})$ over $R_i=\text{Skel}(g_i)$ defined by $t_i=\Pi_{V_i}(t_0)$ where $V_i=\cup_{Z\in R_i}Z$.

Then it is obvious that, for i=1..n, there exists a PSJ-expression $E_i$ total on $R_i$ such that $\mathbf{T}_{E_i}\equiv\mathbf{T}_i$. We also have: ($\delta$) $E=\bowtie_{i=1..n}E_i$.

Then for each i=1..n, $\#(R_i)<p$,by induction hypothesis, ($\gamma$) there exists an expression of $\exists$-V-selection $S_i'\equiv$X:C($\exists(S_i)$) on $g_i$ such that $S_i'$ translates $E_i$.

Consider the following expression of $\exists$-V-selection on f: $S\equiv$X:C(($\exists(S_1)$,...,($\exists(S_n)$). We show that $S$ translates E. For each instance I over f, we have (Lemma 3.4): $S(I)=\circledast_{i=1..n}S'(I[g_i])$.

For each $i,j \in [1..n]$, $i \neq j$, $R_i \cap R_j = X$, thus from the characterization of Verso join in terms of relational operations (Proposition 3.8 , we have:

$\bowtie_{Z \in R} skel(S(I))(Z) = \bowtie_{i=1..n} (\bowtie_{Z \in R_i} skel(S_i'(I[g_i]))(Z))$.

From ($\delta$) and ($\gamma$) and by the characterization of Verso projection in term of relational operation (Theorem 3.9) we obtain that:

$\bowtie_{Z \in R} skel(S(I))(Z) = \bowtie_{1=1..n} E_i(skel(I[g_i])) = E(skel(I))$.

Thus **S** translates E.

□

We are now going to extend the preceding result and characterize PSJ-expression E total on R=Skel(f) such that there exists a Verso expression $\mathcal{E}$ on f composed of a $\exists$-V-selection followed by a Verso projection translating E.

**Lemma 5.3:** Let f be a format and R=Skel(f). Let E be a PSJ-expression total on **R**. Let $\mathbf{T}_E=(t_0,T)$. Then there exists an expression of $\exists$-V-selection $S$ on f and a projected format g of f such that $[g] \circ S$ translates E iff $Max(*(\mathbf{T}_E))=(t_0,\{v_0\})$ where

1. $v_0 \in CRITER(\cup_{Z \in R} Z)$.
2. $t_0 \in CRITER(P)$ and $f_{|P}=g$.

Note that in the above result, $t_0=\Pi_P(v_0)$. The proof of this result is obvious from Lemma 5.2 and the characterization (Theorem 3.9) of Verso projection in terms of relational expression. The proof is omited here.

We now consider PSJ-expression not total on R=Skel(f) and $\exists$-?-V-selection expressions on f. To present our next characterization, we need an intermediate result. Intuitively,

if E is a PSJ-expression on R=Skel(f) and E is not total on R, then there exists a subformat h of f such that E is total on R'=Skel(g). The existence of a ∃-V-selection/Verso projection over h that translates E is then sufficient to deduce the existence of a ∃-?-V-selection/Verso projection on f that translates E.

**Lemma 5.4:** Let f be a format and R=Skel(f). Let E be a PSJ-expression on R. Then there exists h subformat of f such that E is total on R'=Skel(h) and if there exists an expression of ∃-V-selection $S$ on h such that $S$∘[h] translates E then there exists an expression of ∃-?-V-selection $S'$ on f such that [h]∘$S'$ translates E.

**Proof:** We briefly sketch the proof. Let f be a format and g a subformat of f such that $g=f_{|P}$. Let $g_i$, i=1..m be the formats associated with the branches of the tree associated with f and cut by projection of f on P.

Consider any $S$ which is an expression of ∃-V-selection on g such that $S$∘[g] translates E and the ∃-?-V-selection $S'$ obtained by "padding" the expression $S$ with expressions of the form "?($S_{g_i}$)" where $S_{g_i}$ is any expression of Verso simple selection on $g_i$.

Simply using definition 3.2, we show that [g]∘$S'$ translates E. □

From the previous statement, we immediately deduce:

**Lemma 5.5:** Let f be a format and R=Skel(f). Let E be a PSJ-expression on R. Let

$\mathbf{T}_E=(t_0,T)$ . Then there exists an expression of Verso simple selection $S$ on f and a projected format g of f such that $[g]\circ S$ translates E iff $Max(*(\mathbf{T}_E))=(t_0,\{v_0\})$ where

1. $v_0 \in CRITER(\cup_{Z\in R'}Z)$, $R'\subseteq R$ and
2. $t_0 \in CRITER(P)$ and $f_{|P}=g$.

The proof is immediate from Lemma 5.3 and 5.4.

Finally, we enlarge our discussion to encompass the general case. The final result characterizes a PSJ-expression that can be simulated by a Verso selection followed by a Verso projection. (Note here that the Verso selection expressions considered are restricted to the ones containing no "$\not=$" symbols).

**Theorem 5.6:** Let f be a format and R=Skel(f). Let E be a PSJ-expression on R. Let $\mathbf{T}_E=(t_0,T_0)$ . Then there exists an expression of Verso selection $S$ on f and a format g such that $[g]\circ S$ translates E iff $Max(*(\mathbf{T}_E))=(t_0,T)$ where

1. Let $t_0 \in CRITER(P)$, then
   a. $\exists v_0 \in T \mid t_0=\Pi_P(v_0)$
   b. $f_{|P}=g$.
2. $\forall v,w \in T, \exists R'\subseteq R, R'\neq\emptyset \mid \{A|v(A)=w(A)\}=\cup_{Z\in R'}Z$.

Before the presentation of the proof of Theorem 5.6, we propose an example.

**Example 5.7:** Consider the format $f\equiv COURSE(BOOK)^*(STUDENT(GRADE)^*)^*$ and the relational database schema R=Skel(f)={{COURSE},{COURSE,BOOK},

{COURSE,STUDENT},{COURSE,STUDENT,GRADE}}. Consider now the query: "Give the COURSEs in which the STUDENT "toto" is registered and all the STUDENT registered in these COURSEs having obtained a GRADE greater than "10"?"

Typically this query can be expressed over R by the following PSJ-expression:

$$E \equiv \Pi_{COURSE,STUDENT}[select_{GRADE>"10"}[COURSE,STUDENT,GRADE] \bowtie (\Pi_{COURSE}[select_{STUDENT="toto"}[COURSE,STUDENT]])].$$

The schema tableau $\mathbf{T}_E$ associated with E is represented Figure 5-1. Note that $\mathbf{T}_E$ satisfies the condition 1. and 2. of Theorem 5.6. We give below the expression of Verso selection $S$ on f such that [COURSE,STUDENT]$\circ S$ translates E.

$S \equiv$ COURSE : (? (STUDENT : (∃ (GRADE : GRADE>"10"))) |{∃ (STUDENT : STUDENT="toto" (? (GRADE :)))} , ?(BOOK:))

□

| COURSE | BOOK | STUDENT | GRADE |
|---|---|---|---|
| (x1,T) | | (y1,T) | |
| (x1,T)<br>(x1,T) | | (y1,T)<br>(y2,="toto") | (z1,>"10") |

**Figure 5-1:** Schema Tableau associated with E.

In order to proceed to the proof of Theorem 5.6, we need a last intermediate result.

**Lemma 5.8:** Let R be a database shema and $\mathbf{T}=(t_0,T)$ be a shema tableau over R satisfying the properties 1.a, 1.b and 2 of Theorem 5.6. Then the two following statements are equivalent.

1. Max(*(**T**)) satisfies the properties 1.a, 1.b and 2 of Theorem 5.6.
2. There exists $v_0 \in T$ such that the shema tableau $Max(t_0,\{v_0\}\cup\Pi(T))$, denoted $Proj(v_0,\mathbf{T})$, satisfies the properties 1.a, 1.b and 2 of Theorem 5.6.

Intuitively, the preceding result state that the closure under join, the closure under projection and the maximization preserve the properties 1.a, 1.b and 2 of Theorem 5.6. The proof does not present any difficulties and then is omited here.

The proof of Theorem 5.6 is now presented.

**Proof:**

1. We first assume that $[g]\circ S$ translates E. We show that ($\alpha$) $Max(*(\mathbf{T}_E))$ satisfies the conditions 1.a, 1.b and 2. of Theorem 5.6. To simplify the discussion and without loss of generality, we assume in the following that $f\equiv X(f_1)^*$, where $f_1$ is not empty[5]. Then $S$ is of the form $X{:}C(e_1(S_1)|\hat{\mathbf{S}}_1)$. In order to show ($\alpha$), we proceed by an induction on $|\#(R_1)+\#(\hat{\mathbf{S}}_1)|$, where $\#(R_1)$ and $\#(\hat{\mathbf{S}}_1)$ denote the cardinality of $R_1=Skel(f_1)$ and $\hat{\mathbf{S}}_1$ respectively.
   a. if $|\#(R_1)+\#(\hat{\mathbf{S}}_1)|=1$, then $\#(\hat{\mathbf{S}}_1)=0$ and $\#(R_1)=1$. $S\equiv X{:}C\ (e_1\ (S_1)|\emptyset)$ is a Verso simple selection over f and ($\alpha$) is immediat from Lemma 5.3.
   b. Assume now that ($\alpha$) is satisfied for each format f and Verso selection over f such that $|\#(R_1)+\#(\hat{\mathbf{S}}_1)|<p$. Consider f and $S$ such that $|\#(R_1)+\#(\hat{\mathbf{S}}_1)|=p$. Then two cases arise:
      i. $\#(\hat{\mathbf{S}}_1)>0$: Let $\hat{\mathbf{S}}_1=\{\exists(S_1')\}\cup\hat{\mathbf{S}}_a$, $S_a\equiv X{:}C(e_1(S_1)|\hat{\mathbf{S}}_a)$ and $S_b\equiv X{:}C(\exists(S_1'))$. By Lemma 3.5: ($\beta$) $S\equiv S_a \circledast_f ([X]\circ S_b)$.

[5]Note here that Theorem 5.6 is obviously true for $f\equiv X$

By hypothesis, $[g]\circ S$ translates E then $[g]\circ(S_a \circledast_f([X]\circ S_b))$ translates E. On the other hand, there exists $E_a$ and $E_b$ PSJ-expressions over R such that $S_a$ translates $E_a$ , $[X]\circ S_b$ translates $E_b$ and ($\delta$) $E\equiv\Pi_P[E_a \bowtie E_b]$.

$|\#(R_1)+\#(\mathbf{S}_a)|<p$, thus by induction hypothesis, $Max(*(\mathbf{T}_{E_a}))=\mathbf{T}_a=(t_a,T_a)$ satisfies ($\alpha$) and also $Max(*(\mathbf{T}_{E_b}))=\mathbf{T}_b=(t_a,T_a)$ satisfies ($\alpha$).

From ($\delta$), $\mathbf{T}_E$ is equivalent to the tableau $\mathbf{T}=(t,T)$ associated with $\Pi_P[E_a \bowtie E_b]$ and obtained from $\mathbf{T}_a$ and $\mathbf{T}_b$. By definition:

$t=\Pi_P(t_a \bowtie t_b)=\Pi(t_a)$, and

$T=Max(T_a \cup T_b)$.

As $\mathbf{T}_a$ and $\mathbf{T}_b$ satisfy ($\alpha$), it is easy to show that $\mathbf{T}$ satisfies ($\alpha$) and then by Lemma 5.8, we have $Max(*(\mathbf{T}))$ satisfies ($\alpha$).

ii. $\#(\mathbf{S}_1)=0$:The way to prove ($\alpha$) in this case is similar as above. It suffices to decompose $S$. We do not proceed to this step of the proof.

2. Conversly, now assume that E is a PSJ-expression over R such that $Max(*(\mathbf{T}_E))=(t_0,T_0)$ satisfies the conditions 1.a, 1.b and 2. of Theorem 5.6. We show there exists a Verso selection $S$ over f and a subformat g of f such that $[g]\circ S$ translates E.

Let $v_0 \in T_0 \cap CRITER(V_0)$ such that $\Pi_P(v_0)=t_0$ and consider the schema tableau $\mathbf{T}=(t_0,T)=Proj(v_0,Max(*(\mathbf{T}_E)))$. By Lemma 5.8 $\mathbf{T}$ satisfies the properties 1.a, 1.b and 2 of Theorem 5.6 and as $\mathbf{T}$ and $\mathbf{T}_E$ are equivalent, $\mathbf{T}$ defines the same mapping on the URSA-instances over R. In the following, we proceed by induction on $|\#(R)+\#(T)|$.

a. if $|\#(R)+\#(T)|=2$, then $\#(R)=\#(T)=1$ and $T=\{v_0\}$. By Lemma 5.3, there exists a Verso simple selection $S$ and a subformat g of f such that $[g]\circ S$ translates E.

b. Now assume there exists a Verso selection $S$ and a subformat g of f such that $[g]\circ S$ translates E for each PSJ-expression E over R with $|\#(R)+\#(T)|<p$. Let $f\equiv X(f_1)^*...(f_n)^*$. Consider a PSJ-expression E over R=Skel(f) such that $|\#(R)+\#(T)|=p$. Two cases arise:

i. $\#(T)=1$: Again by Lemma 5.3, there exists a Verso simple selection $S$ over f and a subformat g of f such that $[g]\circ S$ translates E.

ii. #(T)>1:Two cases arise: (case 1): there exists $v\in T$, $v\neq v_0$ such that $\{A|v(A)=v_0(A)\}=X$. Let $\mathbf{T_a}=(v_0,T-\{v\})$ and $\mathbf{T_b}=(\Pi_X(v_0),\{v\})$. There exists $E_a$ and $E_b$, PSJ-expressions over R such that $\mathbf{T}_{E_a}\equiv\mathbf{T_a}$ and $\mathbf{T}_{E_b}\equiv\mathbf{T_b}$ and ($\delta$) $E\equiv\Pi_X(E_a\bowtie E_b)$. By construction, $\mathbf{T_a}$ satisfy the conditions 1.a, 1.b and 2. of Theorem 5.6 and thus by Lemma 5.8, Max(*($\mathbf{T_a}$)) does. $\mathbf{T_b}$=Max(*($\mathbf{T_b}$)) satisfies the conditions 1. and 2. of Lemma 5.3 respectively. We have:

($\alpha_1$) $|\#(R)+\#(T-\{v\})|<p$. Then by induction hypothesis , there exists a Verso selection $S_a$ over f such that $[f_{|V_0}]\circ S_a$ translates $E_a$.

Let $S_a\equiv X:C_a(e_{1a}(S_{1a})|\hat{\mathbf{S}}_{\mathbf{1a}},\ldots e_{na}(S_{na})|\hat{\mathbf{S}}_{\mathbf{na}})$.

($\alpha_2$) There exists a Verso simple selection $S_b$ over f such that $[X]\circ S_b$ translates $E_b$. Let $S_b\equiv X:C_b(e_{1b}(S_{1b}),\ldots,e_{nb}(S_{nb}))$.

Note that $C_a=C_b$. Now, using the caracterization of Verso join and projection in terms of relational operations (Proposition 3.8 and Theorem 3.9, we deduce from ($\delta$), ($\alpha_1$) and ($\alpha_2$) that $[f_{|P}]\circ(([f_{|V_0}]\circ S_a)\circledast_{f_{|V_0}}([X]\circ S_b))$ translates E, that is $[f_{|P}]\circ(S_a\circledast_f([X]\circ S_b))$ translates E.

Consider the Verso selection $S\equiv X:C(e_1(S_1)|\hat{\mathbf{S}}_1,\ldots,e_n(S_n)|\hat{\mathbf{S}}_n)$ defined by:

$C=C_a=C_b$,
for i=1..n, $e_i=e_{ia}$, $S_i=S_{ia}$,
for i=1..n, $\hat{\mathbf{S}}_i=S_{ia}\cup\{e_{ib}(S_{ib})\}$

Using Lemma 3.5, we show that $S\equiv S_a\circledast_f[X]\circ S_b$ and then we conclude that $S$ is a Verso selection over f such that $f_{|P}\circ S$ translates E.

(case 2): Assume now there exists no $v\in T$, $v\neq v_0$ such that $\{A|v(A)=v_0(A)\}=X$. Then there exists $j\in[1..n]$ , $\forall v\in T-\{v_0\}$, $v\in CRITER(V)$ implies $V\in Skel(X(f_j)^*)=R_j$. Consider the two tableaux $\mathbf{T_a}=(\Pi_{V_j}(v_0),\{\Pi_{V_j}(v_0)\}\cup T-[v_0])$ where $V_j=\cup_{Z\in R_j}Z$ over $R_j$ and $\mathbf{T_b}=(t_0,\{v_0\})$ over R.

There exits $E_a$, $E_b$ PSJ-expressions over R such that $\mathbf{T}_a \equiv \mathbf{T}_{E_a}$, $\mathbf{T}_a \equiv \mathbf{T}_{E_b}$ and ($\delta$) $E \equiv \Pi_P(E_a \bowtie E_b)$. By construction, $\mathbf{T}_a$ satisfies the conditions 1.a , 1.b and 2. of Theorem 5.6 and thus by Lemma 5.8 Max(*($\mathbf{T}_a$)) does. $\mathbf{T}_b$ satisfies the conditions 1. and 2. of Lemma 5.3.

($\alpha_1$) $|\#(R_j)+\#(T_a)|<p$. Then by induction hypothesis , there exists a Verso selection $S_a$ over $X(f_j)^*$ such that $[f_{|V_{j0}}]\circ S_a$ translates $E_a$ where $V_{j0}=V_j \cap V_0$. Let $S_a \equiv X{:}C_a(e_{ja}(S_{ja})|\hat{\mathbf{S}}_{ja})$.

($\alpha_2$) There exists a Verso simple selection $S_b$ over f such that $[f_{|P}]\circ S_b$ translates $E_b$. Let $S_b \equiv X{:}C_b(e_{1b}(S_{1b}),\ldots,e_{nb}(S_{nb}))$.

It is clear that $C_a=C_b$ and using the charcarterization of Verso join and projection in terms of relational operations (Proposition 3.8 and Theorem 3.9), we deduce from ($\delta$), ($\alpha_1$) and ($\alpha_2$) that $[f_{|P}]\circ(([f_{|V_{j0}}]\circ S_a \circ [f_{|V_j}]) \circledast_f ([f_{|P}]\circ S_b))$ that is $[f_{|P}]\circ((S_a \circ [f_{|V_j}]) \circledast_f (S_b))$ tranlates E.

Consider the Verso selection $S \equiv X{:}C(e_1(S_1)|\hat{\mathbf{S}}_1,\ldots,e_n(S_n)|\hat{\mathbf{S}}_n)$ defined by:

$C=C_a=C_b$,
for i=1..n, $e_i=e_{ib}$ , $S_i=S_{ib}$,
for i=1..n, i$\neq$j, $\hat{\mathbf{S}}_i=\emptyset$ and $\hat{\mathbf{S}}_j=\{e_{ja}(S_{ja})\}$

Using Lemma 3.5, we show that $S \equiv S_a \circ [f_{|V_j}]) \circledast_f (S_b)$ and then we conclude that $S$ is a Verso selection over f such that $f_{|P}\circ S$ translates E.

□

## References

[Abiteboul + Bidoit 84a]
Abiteboul, S. and Bidoit, N.
Non First Normal Form Relations to Represent Hierarchically organized data.
In *Proc. ACM SIGACT-SIGMOD, Waterloo.*, pages 191-198. 1984.

[Abiteboul + Bidoit 84b]
Abiteboul, S. and Bidoit, N.
*Non First Normal Form Relations: An Algebra Allowing Data Restructuring.*
Technical Report 347, Institut National de Recherche en Informatique et Automatique, Centre de Rocquencourt, November, 1984.
submited to Journal of Computer Science and System.

[Aho + Sagiv + Ullman 79a]
Aho, A., Sagiv, Y. and Ullman J.
Efficient Optimization of a Class of Relational Expressions.
*ACM Trans. on Database Systems* 4(4):435-454, 1979.

[Aho + Sagiv + Ullman 79b]
Aho, A., Sagiv, Y. and Ullman J.
Equivalences Among Relational Expressions.
*Siam Journal Comp.* 8(2):218-246, 1979.

[Bancilhon 82] Bancilhon, F. and al.
Verso:A Relational Back-End Data Machine.
In *Proc. Inter., Workshop on Database Machines, San Diego.* 1982.

[Bancilhon + Richard + Scholl 82]
Bancilhon, F., Richard, P. and Scholl, M.
On line Processing of compacted Relations.
In *Proc. Inter. Conf. on VLDB, Mexico*, pages 263-269. 1982.

[Bidoit 84] Bidoit, N.
*Un Modele de Donnees Relationnel Non Normalise: Algebre et Interpretation.*
PhD thesis, Universite de Paris-Sud, Centre d'Orsay, 1984.

[Codd 70] Codd, E.
A relational Model of Data for Large Shared Data Banks.
*CACM* 13(6):377-387, 1970.

[Fisher + Thomas 83a]
Fisher, P. and Thomas, S.
*Nested Relational Structures.*
Technical Report CS-83-09, Department of Computer Science, Vanderbild University, 1983.

[Fisher + Thomas 83b]
Fisher, P. and Thomas, S.
Operations for Non First Normal Form Relation.
In *Proc. IEEE Compsac*, pages 464-475. 1983.

[Jaeschke + Scheck 82]
Jaeschke, C and Scheck, H.
Remarks on the Algebra of Non First Normal Form Relations.
In *Proc. SIGACT-SIGMOD, Los Angeles*, pages 124-138. 1982.

[Klug 80] Klug, A.
*On Inequality Tableaux.*
Technical Report 403, Computer Sciences Department, University of Wisconsin-Madison, November, 1980.

[Kobayashi 80] Kobayashi, I.
*An Overview of Database Management Technology.*
Technical Report TRCS-4-1, Sanno College, Kanagawa 259-11, Japan, 1980.
submited to Journal of Computer Science and System.

[Macleod 83] Macleod, I.
A model for Integrated Information Systems.
In *Proc. VLDB Conf., Florence.* 1983.

[Maier + Mendelzon + Sagiv 79]
Maier, D., Mendelzon, A. and Sagiv, Y.
Testing implications of data Dependencies.
*ACM transactions on Database Systems* 4(4):455-469, 1979.

[Makinouchi 77] Makinouchi, A.
A consideration on Normal Form of Not-Necessarily Normalized Relation in the Relational Model.
In *Proc. Inter. Conf on VLDB, Tokyo* , pages 447-453. 1977.

[Ozsoyoglu 83] Ozsoyoglu, G., and Ozsoyoglu, Z.
An Extension of Relational Algebra for Summary Tables.
In *Proc. 2nd Inter. Conf. on Statistical Database Management, Los Angeles*, pages 202-211. 1983.

[Scheck + Pistor 82]
Scheck, H. and Pistor, P.
Data Structures for an Integrated Data Base Management and Information Retrieval System.
In *Proc. Inter. Conf on VLDB, Mexico*, pages 197-207. 1982.

[Scheck + Scholl 84]
Scheck, H. and Scholl, M.
*An Algebra for the Relational Model with Relatioon-valued attributs.*
Technical Report DVSI-1984-T1, Technische Hochschule Darmstadt, Fachbereit Informatik, 1984.

[Ullman ] Ullman, J.
*Principles of Databases Systems.*
Computer Science Press .

Imprimé en France
par
l'Institut National de Recherche en Informatique et en Automatique