A Rule-Based Language for Ordered Multidimensional Databases*

Mohand-Saïd Hacid

LuFg Theoretical Computer Science
RWTH Aachen, Ahornstraße 55, 52074 Aachen, Germany
hacid@cantor.informatik.rwth-aachen.de

Patrick Marcel, Christophe Rigotti

Laboratoire d'Ingénierie des Systèmes d'Information
INSA Lyon, Bâtiment 501, F-69621 Villeurbanne Cedex
Tél : +33 4 72 43 85 88 - Fax : +33 4 72 43 87 13
{*patrick,crig*}*@lisi.insa-lyon.fr*

Abstract

This paper presents a rule-based language that supports multidimensional tables. It provides a simple and declarative way to express every query computable in polynomial time on ordered tables. We define its model-theoretic semantics and develop an equivalent fixpoint theory that is a basis for the reuse of standard optimization techniques.

1 Introduction

Representation of data in multidimensional structures with direct access (i.e., ordered data structures like lists or multidimensional arrays) is desirable in a variety of disciplines. In the area of databases, Maier and Vance [12] argued that the proposed extensions of the relational model (e.g., nested relations, complex objects, bags, lists, ...) lack in representation capabilities for contexts that could benefit from database technologies (such as scientific computing), and thus they pointed out the need for better supports of ordered data structures. More recently, multidimensional structures has become the central focus of data representations and manipulations used in On-Line Analytical Processing (OLAP) [7, 8], a new challenging technology. Data models and languages capturing some of the OLAP functionalities have been designed [9, 4], however to our knowledge no rule-based language has been proposed.

In this paper we introduce such a language supporting ordered multidimensional tables. It is based on the point of view that a Datalog-like fact represents an entry (called cell reference) in a table. The main problem is to handle simultaneously, at a logical level, the important aspects of multidimensional data representations and manipulations, namely:

*This work is partially supported by Esprit Basic Research Action no. 22469 - Foundations of Data Warehouse Quality.

- the relative position of cells,
- the monovaluation of cells, i.e., the association of a unique cell contents to each cell reference, and
- schema browsing (i.e., table and attribute names used as data).

This difficulty is overcome by combining techniques stemming from previous works done in the area of databases and logic programming:

- the use of a totally ordered domain (e.g., [3, chap. 17]),
- the semantics of "monovaluation" as in Datalog with single-valued data functions [2], and
- a higher-order syntax as in Hilog [6] allowing schema browsing as in F-logic [11].

The main contributions of this paper are the following: first, we define a model-theoretic declarative semantics, that allows a high level specification of multidimensional data manipulations. Next we give a formal equivalent operational semantics that is a basis for the reuse of optimization methods developed for deductive databases. Finally, we show that the resulting language expresses exactly every query computable in polynomial time on ordered multidimensional tables.

This paper is organized as follows. Section 2 introduces the data model. We illustrate through examples the most salient features of the language in Section 3. In Section 4 we give its model-theoretic semantics, and an equivalent fixpoint semantics that leads to a naive evaluation procedure. Section 5 concentrates on the expressive power of the language. We conclude in Section 6.

2 Data Model Overview

In this section, we outline informally the data model underlying our language.

Cells. In our multidimensional data model, data are organized in cells. A cell is identified by a *cell reference*, and is associated with a unique *cell contents*. A cell reference is of the form $N(N_1, N_2, \ldots, N_p)$, where $N, N_1, N_2, \ldots, N_p$ are *names*. N is called the *table* name, and $N_1, N_2, \ldots, N_p$ are called *attribute names*. A cell contents is a tuple of names. Associations of cells contents with cells references are represented by *ground atoms* of the form $N(N_1, N_2, \ldots, N_p) : \langle N_{p+1}, \ldots, N_{p+q} \rangle$.

Tables. A *multidimensional table* (table for short) is a set of ground atoms having a common table name, in which the same reference does not appear more than once to ensure cell monovaluation.

Example 2.1 Consider the description of the box office positions for some films according to their year of release in various countries. One can represent this information in a table called $boxOffice$ as follows:

$\{\ boxOffice(1977, starWars, france) : \langle 1 \rangle,$
$boxOffice(1980, raidersOfTheLostArk, italy) : \langle 103 \rangle\ ,$

$boxOffice(1979, raidersOfTheLostArk, usa) : \langle 1 \rangle$,

$\vdots$

$boxOffice(1975, closeEncounterOfThe3Kind, italy) : \langle 6 \rangle$ }

where, for example, $boxOffice(france, 1977, starWars) : \langle 1 \rangle$ is used to identify a cell containing the box office position of the film Star Wars, for its French year of release, 1977. Fig. 1 shows a graphical representation of the table of Example 2.1. All figures have been gathered in the appendix. □

Database. A *multidimensional database* is a set of ground atoms in which the same reference does not appear more than once.

In order to reflect the specific aspects of multidimensional data representation, our model includes the following features:

- table and attribute names are "reified": they belong to the same domain as cell contents. It provides symmetric treatment to cell references and cell contents,
- structured (*nested*) names can be built from names using the syntactical constructor ".",
- the relative position of the cells is fixed by a total order on the constant of the domain. In our examples, we use the standard lexicographic order (and we assume that digits are lesser than letters, and make no difference between uppercases and lowercases). The order is extended on nested names in a straightforward way.

These features are illustrated in the following examples.

Example 2.2 As the date of release depends functionally on the name of the film and the country, it may seem more convenient to represent the information of example 2.1 by including the date of release of a film within the cell contents. Since attribute names are reified, the following representation can then be used:

{ $boxOffice2(starWars, france) : \langle 1, 1977 \rangle$,

$boxOffice2(raidersOfTheLostArk, italy) : \langle 103, 1980 \rangle$,

$boxOffice2(raidersOfTheLostArk, usa) : \langle 1, 1979 \rangle$,

$\vdots$

$boxOffice2(closeEncounterOfThe3Kind, italy) : \langle 6, 1975 \rangle$ }

A graphical counterpart according to the domain order is given Fig. 2. □

Example 2.3 The film name and its date of release can be grouped together, which gives rise to nested row names:

{ $boxOffice3(starWars \cdot 1977, france) : \langle 1 \rangle$,

$boxOffice3(raidersOfTheLostArk \cdot 1980, italy) : \langle 103 \rangle$,

$boxOffice3(raidersOfTheLostArk \cdot 1979, usa) : \langle 1 \rangle$,

$\vdots$

$boxOffice3(closeEncounterOfThe3Kind \cdot 1975, italy) : \langle 6 \rangle$ }

A graphical counterpart according to the domain order is given Fig. 3. □

It should be noted that our model distinguishes clearly a cell that doesn't exist (i.e., no ground atom with this cell reference in the database), from an existing but empty cell (which is represented by a ground atom of the form $N(N_1, N_2, \ldots, N_p) : \langle\rangle$).

3 Overview of the Rule-Based Language

In this section, we present informally the semantics of the language. We adopt the following conventions: uppercases denote variables, and non capitalized symbols denote constants. We start by giving the intuitive meaning of the deduction rules, and then we present some of the restructuring capabilities of the language.

Intuitive meaning. Consider the rule $p(X) \leftarrow q(X,Y), r(Y)$. The standard (Datalog) informal meaning of this rule is *if q(X, Y) holds and r(Y) holds, then p(X) holds.* The basic intuition of our language is to read such a rule in the following way: *if there are two cells of references q(X, Y) and r(Y), then there is a cell of reference p(X).* We also add the handling of cell contents, and then a typical rule will be: $p(X) : \langle W \rangle \leftarrow q(X,Y) : \langle W \rangle, r(Y) : \langle X \rangle$. This rule will be informally read: *if there exists a cell of reference q(X, Y) containing W, and there exists a cell of reference r(Y) containing X, then there exists a cell of reference p(X) containing W.*

Example 3.1 Consider the representation of the table $boxOffice$ of Fig. 1. The following deduction rule can be used to restructure this table, to obtain the representation of Fig. 2:

$$boxOffice2(F,C) : \langle Y, P \rangle \leftarrow boxOffice(Y,F,C) : \langle P \rangle$$

□

We next present more complex data restructuring, using the table of Fig. 4. This table describes the US box office position of each film, for the five first weeks following their release. This table also contains miscellaneous information: the film genre and the film director. We illustrate first the use of nested names.

Example 3.2 The grouping of films of Fig. 4 along their genre can be expressed by the program:

$$boxOfficeByGenre(G \cdot F, week \cdot W) : \langle P \rangle \leftarrow \leftarrow boxOfficeWeekly(F, week \cdot W) : \langle P \rangle, \\ boxOfficeWeekly(F, misc) : \langle G, D \rangle.$$

$$boxOfficeByGenre(G \cdot F, director) : \langle D \rangle \leftarrow boxOfficeWeekly(F, misc) : \langle G, D \rangle.$$

A representation of table *boxOfficeByGenre* is depicted in Fig. 5. □

Other structurings can be obtained by using cell contents to build table names, in order to restructure data in several tables.

Example 3.3 We split the table *boxOffice* of Fig. 4 into two tables, according to the film director:

$$directedBy \cdot D(F, week \cdot W) : \langle P \rangle \leftarrow boxOffice(F, week \cdot W) : \langle P \rangle, \\ boxOffice(F, misc) : \langle G, D \rangle.$$

$$directedBy \cdot D(F, genre) : \langle G \rangle \leftarrow boxOffice(F, misc) : \langle G, D \rangle.$$

A representation of the corresponding tables are given Fig. 6. □

Now we show how the total order on the constant of the database can be used to manipulate the relative position of cells. We suppose that this order is accessible by means of a table *succ* that contains atoms of the form $succ(a,b) : \langle\rangle$[1]. A cell $succ(a,b) : \langle\rangle$ exists if the

[1] Note that all the cells of table *succ* contain empty tuples.

constant b is the successor of the constant a. In this example, we also use a built-in predicate, noted $\leq$ having its standard meaning.

Example 3.4 We want to find for each film the weeks that contain a position that is a local minimum (i.e., top position). This can be done by the following rule:

$$\begin{aligned} localMinimum(F, week \cdot W_2) : \langle P_2 \rangle \leftarrow\ & boxOfficeWeekly(F, week \cdot W_2) : \langle P_2 \rangle, \\ & succ(W_1, W_2) : \langle\rangle, \\ & succ(W_2, W_3) : \langle\rangle, \\ & boxOfficeWeekly(F, week \cdot W_1) : \langle P_1 \rangle, \\ & boxOfficeWeekly(F, week \cdot W_3) : \langle P_3 \rangle, \\ & P_2 \leq P_1, \\ & P_2 \leq P_3. \end{aligned}$$

To deal with the special cases of $week1$ and $week5$, the following two rules must be added:

$$\begin{aligned} localMinimum(F, week \cdot week1) : \langle P_1 \rangle \leftarrow\ & boxOfficeWeekly(F, week \cdot week1) : \langle P_1 \rangle, \\ & boxOfficeWeekly(F, week \cdot week2) : \langle P_2 \rangle, \\ & P_1 \leq P_2. \end{aligned}$$

$$\begin{aligned} localMinimum(F, week \cdot week5) : \langle P_1 \rangle \leftarrow\ & boxOfficeWeekly(F, week \cdot week5) : \langle P_1 \rangle, \\ & boxOfficeWeekly(F, week \cdot week4) : \langle P_2 \rangle, \\ & P_1 \leq P_2. \end{aligned}$$

A representation of table *localMinimum* is depicted in Fig. 7. □

4 Syntax and Semantics

In this section, we formally present the syntax, and the declarative and operational semantics of the language.

4.1 Syntax

Constants and variables. Let $\mathcal{D}$ be a decidable and totaly ordered set of constants. Let $\leq_{\mathcal{D}}$ be the total order used over $\mathcal{D}$. Let $\mathcal{V}$ be a decidable set of variables, disjoint from $\mathcal{D}$.

Rule-Based Language. We now define the syntactical expressions allowed in the rule-based language:

$$\begin{array}{lcl} name & := & d \mid v \mid name \cdot name \\ value & := & \langle name, \ldots, name \rangle \\ reference & := & name(name, \ldots, name) \\ atom & := & reference : value \\ literal & := & atom \mid \neg atom \\ body & := & literal, \ldots, literal \\ head & := & atom \\ rule & := & head \leftarrow\ body \end{array}$$

where $d \in \mathcal{D}, v \in \mathcal{V}$.

If $n(n_1, \ldots, n_p) : \langle n_{p+1}, \ldots, n_q \rangle$ is an atom, we say that n is the *table name* of the atom. A *positive* literal is an atom. A *negative* literal is a negated atom. In the following, we will note a rule by $A \leftarrow B_1, \ldots, B_n$.

Let *var* be a computable function that assigns to each syntactical expression a subset of $\mathcal{V}$, corresponding to the set of variables occuring in the expression. *var* is extended to sets of expression in a straightforward manner. A ground name (resp. literal, rule) is a name n (resp. literal l, rule r) for which $var(n) = \emptyset$ (resp. $var(l) = \emptyset$, $var(r) = \emptyset$).

Range Restricted Rule. A range restricted rule is a rule $r = A \leftarrow B_1, \ldots, B_n$ where:

- $var(A) \subseteq var(\{B_1, \ldots, B_n\})$,
- let $\mathcal{N}eg$ be the set of negative literals occuring in the body of r, and $\mathcal{P}os$ be the set of positive literals occuring in the body of r. Then $var(\mathcal{N}eg) \subseteq var(\mathcal{P}os)$.

4.2 Semantics

In this section, we give a declarative model-theoretic semantics and an equivalent fixpoint-based operational semantics for programs with a restricted form of negation.

4.2.1 Model-Theoretic Semantics.

Consistency and Interpretation. We note $ref(A)$ the reference part of an atom A. A set I of ground atoms is consistent iff $\forall A_1, A_2 \in I, ref(A_1) = ref(A_2) \implies A_1 = A_2$, where "=" is the syntactical equality. An *interpretation* is a consistent set of ground atoms. This consistency criterion is drawn from the semantics of Datalog with *single-valued data functions* [2].

Remark A Standard stable model semantics and well-founded semantics for normal programs can be generalized for languages with second order syntax and first order semantics like Hilog [14], and can also be used in our case. However, a restricted form of negation in the spirit of semipositive Datalog$^\neg$ is sufficient to express every query on ordered databases computable in polynomial time (see Section 5). Thus, for the sake of simplicity, we choose to adopt this later form. The semantics of a program is given with respect to a finite set of ground atoms called the *input*, that represents the extensional part of the database (as for the presentation of semipositive Datalog$^\neg$ made in [3, Chapter 15]). □

Programs and Inputs. A program P is a pair noted $\langle R_P, edb_P \rangle$ where R_P is a set of range-restricted rules, and edb_P is a set of ground names including *min*, *max* and *succ*. Intuitively, edb_P contains the names of the tables that cannot be populated using rules, but over which negative literals can be used. The tables *min*, *max* and *succ* are used to access the order over $\mathcal{D}$ restricted to the constants manipulated by the program.

Let *table* be a function that assigns to any set of ground literals the set of table names of these literals. An *input* I for a program P is a consistent finite set of ground atoms such that $table(I) \subseteq edb_P$ and I is ordered *wrt* P (defined below).

Let $active_\mathcal{D}$ be the elements of $\mathcal{D}$ that appear in a set of ground literals I and in a program P. Let $\leq_{active_\mathcal{D}}$ be the restriction of $\leq_\mathcal{D}$ to $active_\mathcal{D}$. Then I is said to be ordered *wrt* P if:

1. $min(\alpha) : \langle\rangle \in I \iff \forall \beta \in active_{\mathcal{D}}, \alpha \leq_{active_{\mathcal{D}}} \beta$
2. $max(\alpha) : \langle\rangle \in I \iff \forall \beta \in active_{\mathcal{D}}, \beta \leq_{active_{\mathcal{D}}} \alpha$
3. $succ(\alpha, \beta) : \langle\rangle \in I \iff \beta$ is the immediate successor of α in $active_{\mathcal{D}}$ according to $\leq_{active_{\mathcal{D}}}$.

Valuation. A valuation ν is a total function from $\mathcal{V}$ into $\mathcal{D}$. ν is extended to be the identity on $\mathcal{D}$. ν is also extended in a straightforward manner to names, literals and rules.

Satisfaction. Let $P = \langle R_P, edb_P \rangle$ be a program and $r \in R_P$. Let I be an interpretation. I satisfies r, denoted $I \models r$, iff for each valuation ν, with $\nu(r) = A \leftarrow B_1, \ldots, B_n$ we have:

1. $A \in I$, or
2. $\exists B_i, i \in [1, \ldots, n]$, B_i is a positive literal, and $B_i \notin I$, or
3. $\exists B_i, i \in [1, \ldots, n]$, B_i is a negative literal of the form $\neg C$ and $C \in I$, or
4. $table(\{A\}) \subseteq edb_P$, or
5. $\exists B_i, i \in [1, \ldots, n]$, B_i is a negative literal, and $table(\{B_i\}) \nsubseteq edb_P$.

The cases 1, 2 and 3 reflect the standard semipositive Datalog$^\neg$ semantics. Informally, case 4 garantees that nothing can be stated about cells of tables in edb_P. Case 5 garantees that nothing can be stated using negative literals involving tables not in edb_P.

Model of a Program. An interpretation I is a model of a program $P = \langle R_P, edb_P \rangle$, denoted $I \models P$, if $\forall r \in R_P, I \models r$.

Remark B It should be noticed that even simple programs may have no model, as it is the case in other languages that allow some kind of monovaluation (e.g., Datalog with single-valued data functions [2], COL [1]). As an example, the following program defines two different cell contents for the same cell reference, and thus it has no model:
$a(b, c) : \langle e \rangle \leftarrow$.
$a(b, c) : \langle d \rangle \leftarrow$. □

Remark C We insist on the fact that the valuations map variables of $\mathcal{V}$ only to constants of $\mathcal{D}$. They don't map variables of $\mathcal{V}$ to names constructed with "$\cdot$". This guarantees that if a program admits a model then it admits also a finite model. Consider the following program:
$a(b, c) : \langle e \rangle \leftarrow$.
$a(X \cdot b, c) : \langle e \rangle \leftarrow a(X, c) : \langle e \rangle$.
$\{ a(b, c) : e, a(b \cdot b, c) : e \}$ is a finite model of the program since no valuation can map X to $b \cdot b$. The infinite interpretation $\{ a(b, c) : e, a(b \cdot b, c) : e\ a(b \cdot b \cdot b, c) : e, a(b \cdot b \cdot b \cdot b, c) : e,$ $\ldots \}$ is also a model of this program, but not a minimal one. □

Semantics of a Program. For a program P and an input I for P, the semantics of P on I is, if it exists, the unique minimal model M of P satisfying $M_{|edb_P} = I$, where $M_{|edb_P}$ denotes the restriction of M to atoms which table name belongs to edb_P. This model is denoted $P(I)$.

We can easily prove:

Proposition 4.1 *Let P be a program and I be an input for P. If P admits a model M satisfying $M_{|edb_P} = I$, then $P(I)$ exists and is finite.*

4.2.2 Fixpoint Semantics.

Immediate Consequence Operator. Let $P = \langle R_P, edb_P \rangle$ be a program, and I an interpretation. A ground atom A is an immediate consequence for I and P if either $A \in I$, or $\exists r \in R_P$ and $\exists \nu$ with $\nu(r) = A \leftarrow B_1, \ldots, B_n$, and:

- $\forall i \in [1, \ldots, n]$, if B_i is a negative literal of the form $\neg C$, then $table(\{B_i\}) \subseteq edb_P$, and $C \notin I$, and
- $\forall i \in [1, \ldots, n]$, if B_i is a positive literal, then $B_i \in I$, and
- $table(\{A\}) \nsubseteq edb_P$.

For a program P, we define the immediate consequence operator T_P to be a partial mapping from interpretations of P to interpretations of P, such that, for an interpretation I:

$$T_P(I) = \{A \mid A \text{ is an immediate consequence for } I \text{ and } P\},$$

if this set is consistent; otherwise, $T_P(I)$ is undefined. The following proposition can be established:

Proposition 4.2 *Let P be a program and I an input for P such that $P(I)$ exists, then T_P has a unique minimal fixpoint M satisfying $M_{|edb_P} = I$, which equals $P(I)$.*

Let P be a program and I an input for P, then let

- $T_P^0(I) = I$,
- $T_P^{n+1}(I) = T_P(T_P^n(I))$, if defined.

Using standard techniques, we can prove:

Theorem 4.3 *Let P be a program and I an input for P such that $P(I)$ exists. Then the sequence $\{T_P^i(I)\}_i$ reaches a fixpoint after a finite number N of steps, with $T_P^N(I) = P(I)$.*

Theorem 4.3 provides a straightforward naive evaluation procedure.

5 Expressive Power

In this section, we characterize the expressive power of our language. The result relies on the fact that it is equivalent to that of semipositive Datalog$^\neg$ on ordered databases (see [3] for a presentation).

Proposition 5.1 *Semipositive Datalog$^\neg$ can be simulated within our language.*

Crux Each semipositive Datalog$^\neg$ atom $Q(X_1, \ldots, X_n)$ can be represented in our language by an atom $Q(X_1, \ldots, X_n) : \langle\rangle$. The representation of a semipositive Datalog$^\neg$ program and input is then straightforward.

Proposition 5.2 *Our language can be simulated within semipositive Datalog$^\neg$.*

We illustrate the encoding on the following example:
$R_P = \{T \cdot X(Y) : \langle Z\rangle \leftarrow \ T(X,Y) : \langle Z\rangle, \neg X(Z) : \langle Y\rangle\}$
$edb_P = a, b$
Input $I = \{a(b,c) : \langle b\rangle, min(a) : \langle\rangle, max(c) : \langle\rangle, succ(a,b) : \langle\rangle, succ(b,c) : \langle\rangle\}$
First the table names belonging to edb_P are registred by means of the following Datalog facts:

$$edb\_table\_name(a).$$
$$edb\_table\_name(b).$$

The atoms in I or in R_P are encoded using two special predicate names: idb and edb, and three particular constants: $nest$, att and $cont$. For example, the atom $T \cdot X(Y) : \langle Z\rangle$ is encoded as $idb(T, nest, X, att, Y, cont, Z)$. The predicate name is idb since the atom is the head of the rule. The constant $nest$ indicates that the variable X is nested with the variable T to obtain the table name, att indicates that Y is an attribute, and $cont$ indicates that Z is the cell contents. Following this principle I is encoded as:

$$edb(a, att, b, att, c, cont, b).$$
$$edb(min, att, a).$$
$$edb(max, att, c).$$
$$edb(succ, att, a, att, b).$$
$$edb(succ, att, b, att, c).$$

The encoding of rules reflects the definition of the immediate consequence operator. In this example the rule is encoded:

$$\begin{aligned} idb(T, nest, X, att, Y, cont, Z) \leftarrow\ & edb(T, att, X, att, Y, cont, Z), \\ & \neg edb(X, att, Z, cont, Y), \\ & edb\_table\_name(X), \\ & \neg edb\_table\_name(T, nest, X). \end{aligned}$$

In fact the positive literals in the body of the original rule may hold either in the idb or in the edb part of the semipositive Datalog$^\neg$ program. Thus we also need the following rule:

$$\begin{aligned} idb(T, nest, X, att, Y, cont, Z) \leftarrow\ & idb(T, att, X, att, Y, cont, Z), \\ & \neg edb(X, att, Z, cont, Y), \\ & edb\_table\_name(X), \\ & \neg edb\_table\_name(T, nest, X). \end{aligned}$$

In our language some programs violate cell monovaluation and have no model. They are encoded as semipositive Datalog$^\neg$ programs having a minimal model in which the fact $panic$ holds. To detect cell multivaluation in our example, we need to add the following rule:

$$\begin{aligned} panic \leftarrow\ & idb(T, att, X, att, Y, cont, Z), \\ & idb(T, att, X, att, Y, cont, W), \\ & different(W, Z). \end{aligned}$$

where $different$ is defined by:

$lesserThan(X,Y) \leftarrow edb(succ, att, X, att, Y).$
$lesserThan(X,Y) \leftarrow lesserThan(X,Z), edb(succ, att, Z, att, Y).$
$different(X,Y) \leftarrow lesserThan(X,Y).$
$different(X,Y) \leftarrow lesserThan(Y,X).$

In our langage, cells can be valuated by tuples of various arities. To detect the simultaneous valuation of a cell with tuples of arities 0 and 1, we need to add the following rule:

$panic \leftarrow idb(T, att, X, att, Y),$
$idb(T, att, X, att, Y, cont, Z).$

We can now state the following result:

Theorem 5.3 *Our language expresses exactly* QPTIME.

Proof. By the two previous propositions, and since semipositive Datalog$^\neg$ expresses exactly QPTIME on ordered databases with min and max [13]. □

Remark D Although our programs can be encoded as semipositive Datalog$^\neg$ programs, this encoding is neither natural nor suggestive of specific evaluation technics. Our framework gives a direct syntax and semantics for multidimensionnal table manipulation in deductive databases, and as in the case of Hilog (that can be encoded in Prolog [6]) this direct syntactical and semantical representation of specific concepts provides an easily understandable formulation level and a better basis for evaluation.

□

6 Conclusion

We proposed a rule-based language devoted to multidimensional tables representations and manipulations. Typical examples illustrated its use for restructuring ordered multidimensional databases. We formally defined a model-theoretic semantics and an equivalent fixpoint semantics for this language. It provides a simple and declarative way to express every restructuring operations on ordered multidimensional databases computable in polynomial time, and its naive operational semantics can serve as a basis for the reuse of optimization techniques proposed for deductive databases.

Because of the growing interest in multidimensional models (e.g., OLAP [7]) the corresponding theoretical basis are currently investigated. Gyssens et al. [9] proposed a model of tabular database and an algebra for querying and restructuring it. Agrawal et al. [4] defined an algebra for providing multidimensional manipulations capabilities on top of relational database systems.

To our knowledge, our work is the first one which proposes a formal rule-based language dedicated to multidimensional tabular data manipulations. We described only a core language, and various classical extensions can be made (e.g., the orthogonal combination with a constraint language over a concrete domain [10]). Beyond multidimensional databases

restructuring, a very promising field of application for this language is its use as a data manipulation language for spreadsheet programs. This aspect is discussed in [5]. Our future work concern the incorporation of aggregates to specify table summarization.

References

[1] S. Abiteboul and S. Grumbach. A rule-based language with functions and sets. *ACM TODS*, 16(1):1–30, Mar. 1991.

[2] S. Abiteboul and R. Hull. Data functions, datalog and negation. In *Proc. ACM SIGMOD*, pages 143–153, Chicago, IL, Jun. 1988.

[3] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of databases*. 1995.

[4] R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. Research report, IBM Almaden research center, 650 Harry road, San Jose, CA 95120, 1996.

[5] J.-F. Boulicaut, M.-S. Hacid, P. Marcel, and C. Rigotti. Un langage de manipulation de données pour feuilles de calcul. Research report RR-97-01, LISI, INSA de Lyon, Jan. 1997. 24 pages, in french, submitted.

[6] W. Chen, M. Kifer, and D.S. Warren. HiLog: a foundation for higher-order logic programming. *JLP*, 15(3):187–230, Feb. 1993.

[7] E. F. Codd, S. B. Codd, and C. T. Salley. Providing olap (on-line analytical processing) to user-analysts: An IT mandate. White paper - http://www.arborsoft.com/essabse/wht_ppr/coddTOC.html, 1993.

[8] R. Finkelstein. Understanding the need for on-line analytical servers. White paper - http://www.arborsoft.com/essabse/wht_ppr/finkTOC.html, 1995.

[9] M. Gyssens, L. V. S. Lakshmanan, and I. N. Subramanian. Tables as a paradigm for querying and restructuring. In *Proc. 15th ACM PODS*, Montreal, PQ, Canada, Jun. 1996.

[10] M. S. Hacid, P. Marcel, and C. Rigotti. A rule based CQL for 2 dimensional tables. In V. Gaege, A. Brodsky, O. Günther, D. Srivastava, V. Vianu, and M. Wallace, editors, *Proc. 2nd Int. Workshop on Constraint Database Systems*, volume 1191 of *LNCS*, pages 92–104, Delphi, Greece, Jan. 1997.

[11] M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *JACM*, 42(4):741–843, Jul. 1995.

[12] D. Maier and B. Vance. A call to order. In *Proc. 12th ACM PODS*, pages 1–16, Washington, DC, May. 1993.

[13] C. P. Papadimitriou. A note on the expressive power of prolog. *Bulletin of the EATCS*, 26:21–23, 1985.

[14] K. A. Ross. On negation in hilog. *JLP*, 18(1):27–53, Jan. 1994.

boxOffice

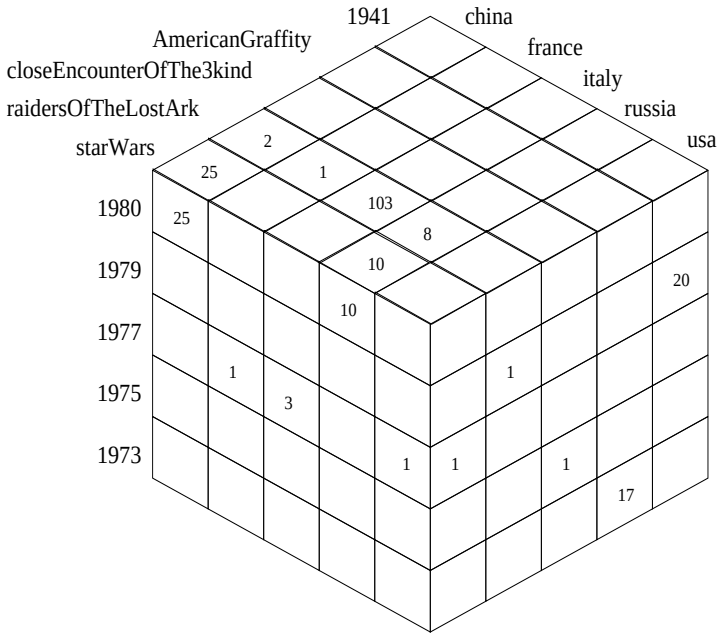

Figure 1: A 3-dimensional representation of the table boxOffice

BoxOffice2	china	france	italy	russia	usa
1941	55,1979	100,1979	121,1979	1,1979	20,1979
americanGraffity	327,1973	5,1973	10,1973	254,1973	17,1973
closeEncounterOfThe3kind	5,1975	6,1975	6,1975	1,1975	1,1975
raidersOfTheLostArk	2,1980	1,1980	103,1980	8,1980	1,1979
starWars	25,1980	1,1977	3,1977	10,1980	1,1977

Figure 2: A 2-dimensional representation of boxOffice

BoxOffice3		china	france	italy	russia	usa
1941	1979	55	100	121	1	20
americanGraffity	1973	327	5	10	254	17
closeEncounterOfThe3kind	1975	5	6	6	1	1
raidersOfTheLostArk	1979					1
	1980	2	1	103	8	
starWars	1977		1	3		1
	1980	25			10	

Figure 3: Another 2-dimensional representation of boxOffice

boxOfficeWeekly	misc	week				
		week1	week2	week3	week4	week5
1941	comedy,spielberg	55	12	10	11	20
americanGraffity	comedy,lucas	27	5	10	25	17
closeEncounterOfThe3kind	scienceFiction,spielberg	5	6	4	1	2
raidersOfTheLostArk	adventures,spielberg	2	1	3	8	10
starWars	scienceFiction,lucas	2	1	3	10	11

Figure 4: Box Office per Weeks

boxOfficeByGenre		director	week				
			week1	week2	week3	week4	week5
adventures	raidersOfTheLostArk	spielberg	2	1	3	8	10
comedy	1941	spielberg	55	12	10	11	20
	americanGraffity	lucas	27	5	10	25	17
scienceFiction	closeEncounterOfThe3kind	spielberg	5	6	4	1	2
	starWars	lucas	2	1	3	10	11

Figure 5: The table boxOfficeByGenre

directedBy lucas	genre	week				
		week1	week2	week3	week4	week5
americanGraffity	comedy	27	5	10	25	17
starWars	scienceFiction	2	1	3	10	11

directedBy spielberg	genre	week				
		week1	week2	week3	week4	week5
1941	comedy	55	12	10	11	20
closeEncounterOfThe3kind	scienceFiction	5	6	4	1	2
raidersOfTheLostArk	adventures	2	1	3	8	10

Figure 6: A table for each director

localMinimum	week				
	week1	week2	week3	week4	week5
1941			10		
americanGraffity		5			17
closeEncounterOfThe3kind	5			1	
raidersOfTheLostArk		1			
starWars		1			

Figure 7: The table localMinimum