A Rule-Based Data Manipulation Language for OLAP Systems*

Mohand-Saïd Hacid[1], Patrick Marcel[2], and Christophe Rigotti[2]

[1] LuFg Theoretical Computer Science
RWTH Aachen, Ahornstraße 55, 52074 Aachen, Germany
hacid@cantor.informatik.rwth-aachen.de
[2] 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 proposes an extension of Datalog devoted to data manipulations in On-Line Analytical Processing (OLAP) systems. This language provides a declarative and concise way to specify the basic standard restructuring and summarizing operations on multidimensional cubes used in these systems. We define its model-theoretic semantics and an equivalent fixpoint semantics that leads to a naive evaluation procedure. We also illustrate its applicability to specify usefull more complex data manipulations arising in OLAP systems.

1 Introduction

On-Line Analytical Processing (OLAP) [7,9] has emerged to support multidimensional data analysis, by providing manipulations and aggregations of data according to multiple dimensions. Codd & al. [7] have proposed an informal model for OLAP in which data are organized in n-dimensional matrices (called *cubes* in OLAP terminology). Systems supporting this model have been developed and several commercial products exist. Foundations of OLAP languages are now a growing field of interest in the database research community. Algebras have been designed [4,11], but to our knowledge no rule-based language has been proposed.

This paper is a contribution in this direction. It introduces an extension of Datalog devoted to the manipulation of data organized in multidimensional cubes. This extension is based on the point of view that a Datalog fact represents an entry (called cell reference) in a cube. The resulting language allows to define intuitively relationships between cells, and provides a declarative way to specify:

1. all the basic cube restructuring operations used in OLAP systems, and
2. complex summarizations of data according to ad-hoc aggregation levels.

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

The semantics of the language has been set up by combining techniques stemming from previous works done in the area of databases and logic programming:

– a higher-order syntax as in Hilog [6] to allow schema browsing;
– a specific relation to specify the aggregation levels, in the spirit of the *isa* hierarchy of F-logic [14];
– the semantics of *monovaluation* as in Datalog with single-valued data functions [2] to ensure the association of each cell reference with an unique contents.

The main contributions of this paper are the following: firstly, we define a model-theoretic declarative semantics that allows a high level specification of the operations used in OLAP systems. Secondly, we give an equivalent constructive semantics, that is a basis for the reuse of evaluation methods developed for deductive databases.

This paper is a major extension of our previous works [13,5,12]. It is organized as follows. Section 2 introduces the multidimensional data model we propose. Motivating examples are given in Section 3 to illustrate informally the most salient features of the data manipulation language. In Section 4 we define its model-theoretic semantics, and an equivalent fixpoint semantics. A variety of complex OLAP manipulations are considered in Section 5. We conclude in Section 6.

2 Data Model Overview

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

Names. The constants we use in the model are called *atomic names.* Structured names can be built from atomic names using the constructor "$\cdot$", and are called *nested names.* In the following, we use the term *names* to denote both atomic and nested names.

Cells. In this 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 *cube* name, and $N_1, N_2, \ldots, N_p$ are called *attribute* names[1]. Given a cell reference $N(N_1, N_2, \ldots, N_p)$, we can view $N_1, \ldots, N_p$ as coordinates in a p-dimensional space.

A cell contents is a tuple of names. Associations of cells contents with cells references are represented by *ground atoms* of the form:

$$N(N_1, N_2, \ldots, N_p) : \langle N_{p+1}, \ldots, N_{p+q} \rangle$$

where the tuple $\langle N_{p+1}, \ldots, N_{p+q} \rangle$ denotes the cell contents. We call this form of atoms *cell-atoms.*

[1] Also called *members* in the OLAP terminology.

Cubes. A *cube* is a set of ground cell-atoms having a common cube name, in which the same reference does not appear more than once to ensure cell monovaluation.

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

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

Example 1. Consider the cube *sales* of Figure 1, which is based on a typical toy example of the OLAP literature [11,16]. The cube *sales* contains information on product sales in areas, over time. In our model, this cube can be represented by:

$$\begin{array}{l} \{sales(nuts, 1996, east) : \langle 50\rangle, \\ \qquad\vdots \\ sales(bolts, 1995, north) : \langle 20\rangle\} \end{array}$$

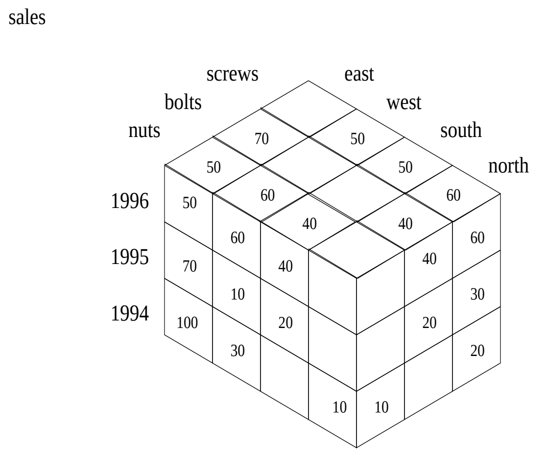

Fig. 1. The cube *sales*

Example 2. A particular 2-dimensional representation of a cube can be obtained by using nested names. Consider the cube *2Dsales* of Figure 2 which represents a nested organization of the data of cube *sales*. Using nested names, this new cube can be described in the model by the set:

$$\begin{array}{l} \{2Dsales(nuts \cdot east, 1996) : \langle 50\rangle, \\ \qquad\vdots \\ 2Dsales(bolts \cdot north, 1995) : \langle 20\rangle\} \end{array}$$

2Dsales		1996	1995	1994
nuts	east	50	70	100
	west	60	10	30
	north			10
	south	40	20	
screws	east		10	10
	west	50	50	50
	north	60	30	20
	south	50	60	60
bolts	east	70	50	40
	west		10	40
	north	40	20	
	south		10	

Fig. 2. A 2-dimensional representation of the cube *sales*

3 Motivating Examples

In this section, we present informally the syntax and the semantics of the language, and show how basic OLAP operations can be expressed.

3.1 Syntax and Semantics

Rules à la Datalog are used to define new cell references and their associated contents from existing cells. The higher order syntax stemming from Hilog [6] allows variables to range over every atomic name used in cell references or cell contents. It should be noted that variables cannot range over nested names but only over atomic names.

We adopt the following conventions: symbols beginning with an upper-case letter denote variables, and symbols beginning with a lower-case letter or a digit denote constants.

Intuitive Meaning. Consider the rule $p(X) \longleftarrow q(X,Y), r(Y)$. 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 extension 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: $p(X) : \langle W \rangle \longleftarrow q(X,Y) : \langle W \rangle, r(Y) : \langle X \rangle$. 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.*

3.2 OLAP Data Manipulations

We now illustrate how rules can specify typical data manipulations used in OLAP systems [11,7,4,16].

Restructuring Cubes. We first show how the basic cube restructuring operations (called *nesting, slicing or dicing, splitting, pivotating or rotating*) can be trivialy specified.

Nesting. The nested representation of Figure 2 can be obtained from the cube of Figure 1 by the rule:

$$2Dsales(P \cdot R, Y) : \langle S \rangle \longleftarrow \; sales(P, Y, R) : \langle S \rangle.$$

Slicing - Dicing. Suppose now we are interested in a subset (a slice) of the cube *sales*, that contains only the data for year 1996. One can define a corresponding 2-dimensional cube from *sales* by the rule:

$$sales \cdot 1996(P, R) : \langle S \rangle \longleftarrow \; sales(P, 1996, R) : \langle S \rangle.$$

A graphical counterpart of cube $sales \cdot 1996$ is represented in Figure 3.

sales 1996	east	west	north	south
nuts	50	60		40
screws		50	60	50
bolts	70		40	

Fig. 3. The cube $sales \cdot 1996$

Splitting. Suppose we want to split the cube *sales* in order to obtain a cube per region. We can use a variable ranging over regions to form new cube names. This is shown by the following rule:

$$sales \cdot R(P, Y) : \langle S \rangle \longleftarrow \; sales(P, Y, R) : \langle S \rangle.$$

Graphical counterparts of cubes $sales \cdot east$, $sales \cdot west$, $sales \cdot south$ and $sales \cdot north$ are represented in Figure 4.

sales east	1996	1995	1994
nuts	50	70	100
screws		10	10
bolts	70	50	40

sales west	1996	1995	1994
nuts	60	10	30
screws	50	50	50
bolts		10	40

sales south	1996	1995	1994
nuts	40	20	
screws	50	60	60
bolts		10	

sales north	1996	1995	1994
nuts			10
screws	60	30	20
bolts	40	20	

Fig. 4. A cube per region

Pivoting - Rotating. The following rule can be used to transpose the cube $sales \cdot 1996$:

$$transposed(R,P) : \langle S \rangle \longleftarrow \; sales \cdot 1996(P,R) : \langle S \rangle.$$

A graphical counterpart of cube *transposed* is represented in Figure 5.

transposed	nuts	screws	bolts
east	50		70
west	60	50	
north		60	40
south	40	50	

Fig. 5. The cube *transposed*

Summaries Specification. We now present how rules can be used to specify summarizations of data at different levels of aggregation. The following examples are based on a 2-dimensional cube named $sales \cdot byCity$, that contains information on product sales in various cities (Figure 6). The specification of summaries requires that the grouping relationship between attribute names is known (e.g., how cities are grouped in regions). This relationship represents what is called *hierarchies* in the OLAP literature. Such a relationship is depicted in Figure 7. In this example, cities can be grouped in regions, and regions can be grouped to form the whole *area*. In the same way, the different products can be grouped to form the whole production called *product*.

sales byCity	bordeaux	dijon	grenoble	lille	lyon	marseille	montpellier	nantes	paris	poitier
nuts	50	60		40	70	50	80		20	60
screws		50	60	50		50	60	50		60
bolts	70		40	100	50	60		80	40	

Fig. 6. The cube $sales \cdot byCity$

The grouping relationship can be specified by rules. This relationship is represented by particular literals of the form $in(\alpha, \beta)$, where α and β are atomic names. For example the grouping relationship of Figure 7 is described by the following set of ground grouping atoms:

$$\{in(nuts, product), \ldots, in(marseille, south), \ldots, in(south, area)\}$$

Aggregate subgoals Aggregate subgoals used to specify the summaries are of the form: $T = f(N(N_1, \ldots, N_p))$ where T is a constant or a variable, f is an

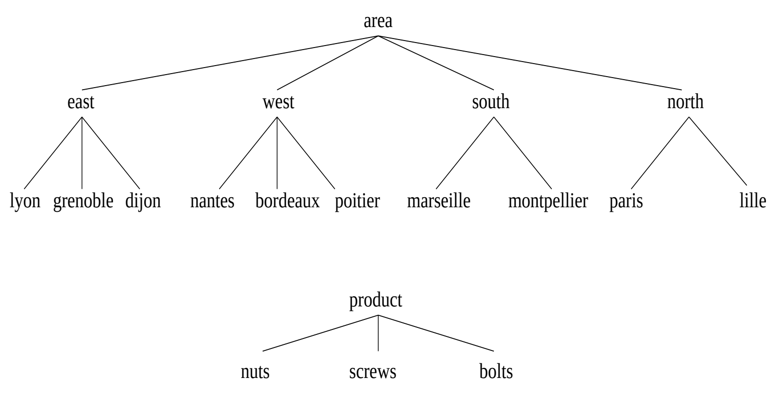

Fig. 7. The product and area grouping hierarchies

aggregate operator (e.g., *sum, min*) and $N(N_1, \ldots, N_p)$ is a possibly non ground cell reference.

Their intuitive meaning is illustrated by the following example. Consider the ground aggregate subgoal $130 = sum(sales \cdot byCity(nuts, east))$. It holds if the sum of the sales of nuts over the cities *in* region *east* is equal to 130.

More precisely, consider a set noted $detailRef(sales{\cdot} byCity(nuts, east))$ that contains the references of the existing cells corresponding to the lowest level of description for the sales of nuts in region east. According to the grouping relationship depicted in Figure 7, we have:

$$detailRef(sales \cdot byCity(nuts, east)) = \\ \{sales{\cdot} byCity(nuts, lyon), sales{\cdot} byCity(nuts, dijon)\}.$$

It should be noticed that $sales \cdot byCity(nuts, grenoble)$ does not belong to this set, since no cell with this reference exists in the database.

Consider $detailCont(sales{\cdot} byCity(nuts, east))$ defined as the multiset formed with the cell contents of the cell references in $detailRef(sales{\cdot} byCity(nuts, east))$. We have:

$$detailCont(sales \cdot byCity(nuts, east)) = \{\langle 70\rangle, \langle 60\rangle\}.$$

The semantics of the aggregate subgoal can now be stated more precisely: $130 = sum(sales \cdot byCity(nuts, east))$ holds if the sum of the elements in $detailCont(sales \cdot byCity(nuts, east))$ is equal to 130.

It should be noticed that $detailRef$ collects the most detailed existing description according to all coordinates. Consider for example the set $detailRef(sales \cdot byCity(product, area))$. We have:

$$\begin{array}{l} detailRef(sales \cdot byCity(product, area)) = \\ \quad \{sales \cdot byCity(nuts, bordeau), sales \cdot byCity(nuts, dijon), \\ \quad \quad \vdots \\ \quad sales \cdot byCity(bolts, nantes), sales \cdot byCity(nuts, paris)\}. \end{array}$$

Hence $sum(sales \cdot byCity(product, area)$ denotes the total of sales for the whole production over the whole area.

Rolling-up. Suppose we want to roll-up[2] the sales from cities to the whole area, to obtain a cube containing the global sales for each product over the whole area. This can be done with the rule:

$$globalSales \cdot byProduct(P, total) : \langle T \rangle \longleftarrow \begin{array}{l} T = sum(sales \cdot byCity(P, area)), \\ in(P, product). \end{array}$$

A graphical counterpart of cube $globalSales \cdot byProduct$ is represented in Figure 8.

globalSales byProduct	total
nuts	430
screws	380
bolts	440

Fig. 8. The cube $globalSales \cdot byProduct$

Rolling-up from Cities to Regions. If we do not want to roll-up to the whole area, but only to the different regions, this can be specified by using a variable ranging over the region names:

$$sales \cdot byRegion(P, R) : \langle T \rangle \longleftarrow \begin{array}{l} T = sum(sales \cdot byCity(P, R)), \\ in(P, product), in(R, area). \end{array}$$

A graphical counterpart of cube $sales \cdot byRegion$ is represented in Figure 9.

sales byRegion	east	west	south	north
nuts	130	110	130	60
screws	110	110	110	50
bolts	90	150	60	140

Fig. 9. The cube $sales \cdot byRegion$

Multiple Roll-up. To obtain the global sales by region, we can roll-up simultaneously from the product names to the whole production, and from the cities to the regions:

$$globalSales \cdot byRegion(R, total) : \langle T \rangle \longleftarrow \begin{array}{l} T = sum(sales \cdot byCity(product, R)), \\ in(R, area). \end{array}$$

[2] Roll-up is synonymous with *consolidate* or *summarize* in the OLAP literature.

A graphical counterpart of the cube *globalSales* · *byRegion* is represented in Figure 10.

globalSales · byRegion	total
east	330
west	370
south	300
north	250

Fig. 10. The cube *globalSales* · *byRegion*

4 Syntax and Semantics

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

4.1 Syntax

Constants and variables. Let $\mathcal{D}$ be a decidable set of constants called atomic names, and $\mathcal{V}$ be a decidable set of variables, disjoint from $\mathcal{D}$.

Aggregates An aggregate operator f is a partial mapping from multisets of tuples over $\mathcal{D}$ to a single value in $\mathcal{D}$. Let $\mathcal{AGG}$ be a set of aggregate operators.

Rule-Based Language. The syntactical expressions allowed in the rule-based language are:

$$\begin{array}{ll}
atomicName & := c \mid v \\
name & := atomicName \mid name \cdot name \\
contents & := \langle name, \ldots , name\rangle \\
reference & := name(name, \ldots , name) \\
cell - atom & := reference : contents \\
groupingAtom & := in(atomicName, atomicName) \\
atom & := cell - atom \mid groupingAtom \\
aggregateSubgoal & := atomicName = f(reference) \\
literal & := atom \mid aggregateSubgoal \\
body & := literal, \ldots , literal \\
head & := atom \\
rule & := head \longleftarrow body
\end{array}$$

where $c \in \mathcal{D}$, $v \in \mathcal{V}$, and $f \in \mathcal{AGG}$.

Definitions. We note $ref(A)$ the reference part of a cell-atom or aggregate subgoal A, and $val(A)$ the contents part of a cell-atom A. Let var be a computable function that assigns to each syntactical expression the subset of $\mathcal{V}$ corresponding to the set of variables occuring in this expression. var is extended in a straightforward manner to sets of syntactical expressions. A ground name (resp. reference, literal) is a name n (resp. reference rf, literal l) for which $var(n) = \emptyset$ (resp. $var(rf) = \emptyset$, $var(l) = \emptyset$).

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

- $var(A) \subseteq var((\{B_1, \ldots, B_n\}))$, and
- let $\mathcal{A}g$ be the set of references occuring in aggregate subgoals in r, and $\mathcal{B}$ be the set of atoms occuring in the body of r, then $var(\mathcal{A}g) \subseteq var(\mathcal{B})$.

Restructuring Programs. A restructuring program is a set of range restricted rules having *no aggregate subgoal in their body.*

Summarizing Programs. A summarizing program is a set of range restricted rules having *no grouping atom in their head.*

OLAP Programs. An OLAP program is a pair $\langle R, S\rangle$, where R is a restructuring program, and S is a summarizing program. Informally, the semantics of an OLAP program $\langle R, S\rangle$ will be to consider R and S in two stages. The restructuring program R will be used first, and then S will be applied.

4.2 Semantics

In this section, we give a declarative model-theoretic semantics and an equivalent fixpoint-based constructive semantics.

Input. The semantics of a program (restructuring, summarizing or OLAP program) is given with respect to a set of ground atoms describing the *extensional* part of the database. Following [3], we call this set an *input.*

Model-Theoretic Semantics.

Grouping Relationship. Let J be a set of ground atoms. Then in_J is the relation corresponding to the description of the grouping atoms within J. More precisely, we define the relation in_J by: $\forall x, y \in \mathcal{D}, in_J(x, y) \iff in(x, y) \in J$.

Extending the Grouping Relationship Over References. Let J be a set of ground atoms. We define a relation noted $<_J$ over references in the following way: for all references rf and rf', with $rf = n(n_1, \ldots, n_p)$ and $rf' = n(n'_1, \ldots, n'_p)$, then $rf <_J rf' \iff rf \neq rf'$ and $\forall i \in [1, \ldots, p]$, either $in_J(n_i, n'_i)$ holds, or $n_i = n'_i$.

Informally, we have $rf <_J rf'$ when rf references a cell at a strictly lower level of detail than rf', with respect to the grouping relationship defined in J.

Interpretation with respect to an input. Let I be an input. A set J of ground atoms is an *interpretation* with respect to I if the following conditions are satisfied:

- for all A_1, A_2 cell-atoms in I, $ref(A_1) = ref(A_2) \implies A_1 = A_2$, where "=" is the syntactical equality. This criterion is drawn from the semantics of Datalog with *single-valued data functions* [2]. It garantees that a cell is associated with an unique cell contents;
- the transitive closure of in_J is irreflexive. This ensures that the grouping relationship has no directed cycle;
- $I \subseteq J$ (i.e., if something holds in the input, it also holds in the interpretation).

Most Detailed Information. The satisfaction of a ground aggregate subgoal of the form $k = f(n(n_1, \ldots, n_p))$ depends on the most detailed information available for $n(n_1, \ldots, n_p)$ *in the input.* This is formalized by two functions: *detailRef* and *detailCont*. $detailRef(n(n_1, \ldots, n_p))$ is the set of cell references of the most detailed information available for $n(n_1, \ldots, n_p)$ in the input, and $detailCont(n(n_1, \ldots, n_p))$ is the multiset of the contents of these cells. More formally: let J be an interpretation with respect to an input I. Let B be a ground aggregate subgoal. We define the set $detailRef_I^J(B) = \{A \in I \mid ref(A) <_J ref(B)\}$, and the multiset $detailCont_I^J(B) = \{k \mid k = val(A), A \in detailRef_I^J(B)\}$.

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 J be an interpretation with respect to an input I. J satisfies a ground literal B with respect to I, denoted $J \models_I B$, iff

- B is a ground atom and $B \in J$, or
- B is a ground aggregate subgoal of the form $k = f(n(n_1, \ldots, n_p))$, such that $detailRef_I^J(B) \neq \emptyset$, $f(detailCont_I^J(B))$ is defined, and $f(detailCont_I^J(B)) = k$.

Remark 3. It should be noticed that our semantics restricts the use of aggregate subgoals to the input. This simplifies the semantics but limits the expressivity of a summarizing program. However, in the case of an OLAP program of the form $\langle R, S \rangle$, the input of the summarizing part S will be specified by the restructuring program R.

Let $r = A \longleftarrow B_1, \ldots, B_n$ be a rule, and I be an input. An interpretation J satisfies r with respect to I, denoted $J \models_I r$ iff for each valuation ν we have:

- $J \models_I \nu(A)$, or
- $\exists B_i, i \in [1, \ldots, n], J \not\models_I \nu(B_i)$.

Model of a Restructuring or Summarizing Program. An interpretation J is a model of a restructuring or summarizing program P with respect to an input I, denoted $J \models_I P$, iff $\forall r \in P, J \models_I r$.

Remark 4. 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:

$a(b, c) : \langle e \rangle \longleftarrow \ .$
$a(b, c) : \langle d \rangle \longleftarrow \ .$

Remark 5. Inconsistency may also arise from the specification of the grouping relationship. As in other languages that handle some kind of hierarchy (e.g., the *subclass* relationship in F-logic [14]), a program like:

$in(a, b) \longleftarrow \ .$
$in(b, a) \longleftarrow \ .$

will have no model, since it specifies a directed cycle in the grouping relationship.

Remark 6. 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:

$a(b, c) : \langle e \rangle \longleftarrow \ .$
$a(X \cdot b, c) : \langle e \rangle \longleftarrow \ a(X, c) : \langle e \rangle.$

$\{\ a(b, c) : \langle e \rangle,\ a(b \cdot b, c) : \langle e \rangle\ \}$ is a finite model of the program since no valuation can map X to $b \cdot b$. The infinite interpretation $\{\ a(b, c) : \langle e \rangle,\ a(b \cdot b, c) : \langle e \rangle\ a(b \cdot b \cdot b, c) : \langle e \rangle,\ a(b \cdot b \cdot b \cdot b, c) : \langle e \rangle, \ \ldots\ \}$ is also a model of this program, but not a minimal one.

Semantics of a Restructuring or Summarizing Program. For a restructuring or summarizing program P and an input I, the semantics of P on I is the unique minimal model of P with respect to I, if it exists. It is denoted $P(I)$.

Using standard techniques, we can prove:

Proposition 7. *Let P be a restructuring or summarizing program and I be an input. If P admits a model, then $P(I)$ exists and is finite.*

Semantics of an OLAP Program. Let $Q = \langle R, S \rangle$ be an OLAP program, and I be an input. The semantics of Q on I is $S(R(I))$ if it exists. It is denoted $Q(I)$.

Fixpoint Semantics.

Immediate Consequence Operator. Let P be a restructuring or summarizing program, I be an input, and J be an interpretation with respect to I. A ground atom A is an immediate consequence for J and P with respect to I if either $A \in J$, or $\exists r \in P$ and $\exists \nu$ with $\nu(r) = A \longleftarrow B_1, \ldots, B_n$, such that $\forall i \in [1, \ldots, n], J \models_I B_i$.

For a restructuring or summarizing program P and an input I, we define the immediate consequence operator $T_{P,I}$ to be a partial mapping from interpretations of P to interpretations of P with respect to I, such that, for an interpretation J:

$T_{P,I}(J) = \{A \mid A$ is an immediate consequence for J and P with respect to $I\}$
if this set is an interpretation with respect to I;
otherwise, $T_{P,I}(J)$ is undefined.

The following proposition can be established:

Proposition 8. *Let P be a restructuring or summarizing program and I be an input such that $P(I)$ exists. Then $T_{P,I}$ has a unique minimal fixpoint, that equals $P(I)$.*

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

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

Using standard techniques, we can prove:

Proposition 9. *Let P be a restructuring or summarizing program and I an input such that $P(I)$ exists. Then the sequence $\{T_P^i(I)\}_i$ reaches a fixpoint after a finite number N of steps, with $T_P^N(I) = P(I)$.*

Let $Q = \langle R, S \rangle$ be an OLAP program and I be an input. Proposition 9 defines a constructive semantics for $R(I)$, and then for $S(R(I))$ (i.e., $Q(I)$). This provides a straightforward naive evaluation procedure.

5 Applications of the Language

In this section, we illustrate the applicability of our language to specify in a clear and concise way various complex manipulations [8,4,7,10,15] arising in OLAP applications (*push-pull, drill-down, ad-hoc grouping* and the so-called *cube operator*).

5.1 Description of the Database

We use in our examples a restricted form of the database described in the OLAP Benchmark APB-1 [8]. It consists in a cube named $c1$, that contains the information required by a supplier to analyze product sales to customers over time. The sales information are units sold and dollar sales. In our model, we use cell-atoms of the form $c1(m, p, st) : \langle us, ds\rangle$ to represent the units sold (us) and dollar sales (ds) for a product p sold to a store st on month m. So the cube $c1$ contains the monthly results (units sold and dollar sales) of products (represented by codes) sold to stores for a whole year. The possible groupings described by the relation *in* are:

– stores can be grouped by retailers, and retailers can be grouped to form the whole distribution to customers, called *customer*;
– the different products can be grouped to form the whole production called *product*;
– the months can be grouped to form the whole year, called *time-period*.

5.2 Manipulations

In this section, the rules of the restructuring (resp. summarizing) part of the OLAP program will be noted with an arrow of the form $\overset{r}{\longleftarrow}$ (resp. $\overset{s}{\longleftarrow}$).

Push - Pull. The operations of pushing and pulling [4] allow meta-data (cell references) and data (cell contents) to be treated uniformally:

– pushing the period in the cell contents:
$$c2(M, P, St) : \langle US, DS, M\rangle \overset{r}{\longleftarrow} c1(M, P, St) : \langle US, DS\rangle$$
– pulling the units sold in the cell reference:
$$c3(M, P, St, US) : \langle DS\rangle \overset{r}{\longleftarrow} c1(M, P, St) : \langle US, DS\rangle$$

For the sake of simplicity, we use in the rest of the section the cube $c4$ containing only monthly dollar sales per store. $c4$ is obtained by:
$$c4(M, P, St) : \langle DS\rangle \overset{r}{\longleftarrow} c1(M, P, St) : \langle US, DS\rangle$$

Roll-up to All Possible Levels. A single rule can specify the summaries of the sales in cube $c4$ at all grouping levels. Moreover, the resulting summaries can be placed within the cube $c4$ itself. Suppose we want to include into the cube $c4$ for each product P, the summaries of the *january* sales at two different levels: at the retailer level, and at the global level grouping all customers. This can be done with the single rule:
$$\begin{aligned} c4(january, P, C) : \langle S\rangle \overset{s}{\longleftarrow}\ & S = sum(c4(january, P, C)),\\ & in(P, product),\\ & in(X, C). \end{aligned}$$
Each value of C according to each instantiation of $in(X, C)$ gives rise to a different grouping. Hence, summaries are specified at all grouping levels for customers.

Drill-down. Navigation towards more detailed data [7], called *drill-down* in the OLAP literature, can also be specified by rules. Consider the cube $c4$ extended with the cells specified by the previous rule. Suppose that having filtered the sales by retailer greater than 100 in january, we want to extract the corresponding sales at a more detailed level (i.e., the store level). This can be expressed by the following rule, where $\geq$ is a built-in predicate having its standard meaning:

$$\begin{aligned} retailerDetails \cdot january(St, P) : \langle DS \rangle \overset{s}{\longleftarrow}\ & c4(january, P, R) : \langle S \rangle, \\ & S \geq 100, \\ & in(R, customer), \\ & in(P, product), \\ & c4(january, P, St) : \langle DS \rangle, \\ & in(St, R). \end{aligned}$$

It should be noticed that this rule must be in the summarizing part of the OLAP program since its intended meaning is to use the summaries specified by the previous rule.

Ad-hoc Grouping [7]. The grouping relationship can be specified by rules, in order to define specific summaries. Consider a new cube named *responsible*, in which a cell of reference $responsible(P, St)$ contains the name of the person responsible for a product P in a store St (a person may be responsible for several products in several stores). For each person, we want to summarize the sales concerning the products he is responsible for in january (according to the products and the stores).

First, we define how products are grouped by responsible names:

$$in(P, R) \overset{r}{\longleftarrow} responsible(P, St) : \langle R \rangle.$$

And we specify the way stores may be grouped by responsibles:

$$in(St, R) \overset{r}{\longleftarrow} responsible(P, St) : \langle R \rangle.$$

Then we simply summarize the sales for each responsible names appearing in the cube responsible:

$$\begin{aligned} result(R, totalSales) : \langle S \rangle \overset{s}{\longleftarrow}\ & S = sum(c4(january, R, R)), \\ & responsible(X, Y) : \langle R \rangle. \end{aligned}$$

Cube Operator. The cube operator proposed by Gray & al. [10] is a n-dimensional generalization of the SQL group by operator. Intuitively, the query:

$$\begin{aligned} & select\ \ldots \\ & from\ \ldots \\ & cube\ by\ a, b, c \end{aligned}$$

computes the group by aggregates for each possible subset of $\{a, b, c\}$. This operator has in turn been generalized by Shukla & al [15] towards cubes with aggregation hierarchies. In this case, the cube operator computes the group by aggregates for each possible subset of the given set of attributes at each possible *level* of aggregation in the hierarchies.

On the cube $c4$, this operation can simply be specified by:

$$\begin{aligned} cubeOperatorResult(T, P, C) : \langle S \rangle \overset{s}{\longleftarrow}\ & S = sum(c4(T, P, C)), \\ & level(T) : \langle\rangle, \\ & level(P) : \langle\rangle, \\ & level(C) : \langle\rangle. \end{aligned}$$

where *level* may be defined by the two following rules:

$$\begin{aligned} & level(X) : \langle\rangle \overset{r}{\longleftarrow} in(X, Y). \\ & level(Y) : \langle\rangle \overset{r}{\longleftarrow} in(X, Y). \end{aligned}$$

6 conclusion

We proposed an extension of datalog devoted to the specification of data manipulations in OLAP systems. We formally defined its model-theoretic semantics, an equivalent fixpoint semantics, and we illustrated on typical examples its capabilities of handling standard restructuring and summarizing operations on multidimensional cubes.

Because of the growing interest in OLAP systems, the foundations of their data manipulation languages are currently investigated. Gyssens et al. [11] proposed a model of tabular database and developed an algebra for querying and restructuring tabular information. Agrawal et al. [4] defined an algebra for providing multidimensional manipulations capabilities on top of relational database systems. Specific grouping operators have also been designed [10,15], but to our knowledge, no rule-based language has been proposed.

We present such a language, having two main interests. Firstly, it allows a clear and concise specification of complex operations used in OLAP systems. And secondly, it can serve as a basis for the reuse in these systems of the optimization methods developed for deductive databases.

Our previous works have investigated several related aspects:

- [12] focuses on the manipulations of ordered multidimensional databases;
- [13] presents an orthogonal combination with a constraint language over a concrete domain;
- [5] discusses the use of such a rule-based language as a data manipulation language for spreadsheet programs.

This paper concentrates on OLAP systems, and is a major extension of our previous works to handle the specification of summarizing operations. Future works include investigations of efficient evaluation techniques based on those used in classical deductive database systems.

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. In *Proc. 13th ICDE*, Birmingham, United Kingdom, Apr. 1997.
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/essbase/wht_ppr/coddTOC.html, 1993.

8. The OLAP Council. Apb-1 benchmark. http://www.olapcouncil.org/research /bmarkly.htm, 1997.
9. R. Finkelstein. Understanding the need for on-line analytical servers. White paper - http://www.arborsoft.com/essbase/wht_ppr/finkTOC.html, 1995.
10. J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube : a relational operator generalizing group-by, cross-tab, and sub-totals. In *Proc. 12th ICDE*, pages 152–159, New Orleans, LO, Feb. 1996.
11. 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.
12. M. S. Hacid, P. Marcel, and C. Rigotti. Extending datalog for ordered multidimensional databases. In *Proc. 5th Int. Workshop on Deductive Database and Logic Programming (DDLP'97)*, Leuven, Belgium, Jul. 1997. To appear.
13. 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 (CDB'97)*, volume 1191 of *LNCS*, pages 92–104, Delphi, Greece, Jan. 1997.
14. M. Kifer, G. Lausen, and J. Wu. Logical foundations of object-oriented and frame-based languages. *JACM*, 42(4):741–843, Jul. 1995.
15. A. Shukla, P. M. Deshpande, J. F. Naughton, and K. Ramasamy. Storage estimation for multidimensional aggregates in the presence of hierarchies. In *Proc. 22nd VLDB*, pages 522–531, Bombay, India, Sep. 1996.
16. Pilot Software. An introduction to olap, 1995. http://www.pilotsw.com/r_and_t /whtpaper/olap/olap.htm.