Proceeding 6th Int. Workshop on Deductive Databases and Logic Programming DDLP'98,
Manchester (UK), June 20, 1998,
P. Fraternali, U. Geske, C. Ruiz and D. Seipel (Editors)

Spreadsheet generation from rule-based specifications

Jean-François Boulicaut

University of Helsinki
Department of Computer Science, P.O. Box 26
FIN-00014 University of Helsinki, Finland
boulicau@cs.helsinki.fi

Patrick Marcel, François Pinet, Christophe Rigotti

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

Abstract

Experimental studies have pointed out that operational spreadsheets contain a lot of errors. In this paper, we propose to use a multidimensional rule-based data manipulation language to specify spreadsheets. This language allows formal declarative definition of spreadsheet tabular data treatments. The main contribution of this paper is to show that from textual rule-based specifications, we can generate the corresponding spreadsheet. This generation of a spreadsheet from its logical definition is a promising way to reduce encoding errors. The technique presented has been implemented in a crude running prototype.

1 Introduction

Spreadsheet softwares and their user-friendly interface have become a very popular tool for data presentation and analysis. Today's use of spreadsheets encompasses both quick formulation of simple problems and powerful front-ends to business decision support systems like On-Line Analytical Processing (OLAP) [7].

Unfortunately, spreadsheets still lack of software engineering tools to control and maintain quality criteria like soundness or readability. Experimental studies have pointed out a high rate of errors in operational spreadsheets and its increasing economical impact [5, 8, 18]. Using database terminology, it has been advocated [14, 13] that such errors are made by confusing the spreadsheet physical level (i.e., the grid) with its logical level (i.e, the tables definition).

This paper proposes to rely on database concepts to provide spreadsheets with a formal logical level, in order to logically control *a priori* the spreadsheets quality. We show that a rule-based Data Manipulation Language designed to handle a multidimensional data model can be used to formally define spreadsheet tabular data manipulation.

The language we use and its associated data model stems from our previous works done in the area of rule-based languages for multidimensional databases [9, 11, 10, 12]. It is based on the point of view that a Datalog fact represents an entry (called cell reference) in a multidimensional table. Therefore it allows intuitive definitions of relationships between cells, and provides a declarative way to specify table querying and restructuring. Formal semantics for this language has been set up by combining techniques stemming from previous works done in the area of databases and logic programming [6, 1] and can be found in [9, 11]. The specification of spreadsheets at a logical level has already been envisaged in [3, 4] but has not been published yet.

The main contribution of this paper is to present a method for generating the physical level of a spreadsheet from its logical definitions. The following section describes the language and illustrates on few examples its suitability to handle spreadsheet data and treatments. In section 3 we present on an example our generation technique. We conclude in section 4.

2 Logical Level Overview

In this section, we give a brief and informal presentation of the rule-based language and its associated multidimensional data model. For the sake of space, we redirect the interested reader towards [9, 11, 3, 12] for formal presentations, various extensions and more examples. We adopt standard conventions: symbols beginning with an upper-case letter denote variables, and symbols beginning with a lower-case letter or a digit denote constants.

2.1 Data Model

In our 2-dimensional tabular data model, data are organized in *cells*. Cells can be seen as logical counterparts to physical squares of the spreadsheet's grid. A cell is identified by a *cell reference*, and is associated with a unique *cell contents*. A cell reference is of the form $T(R, C)$, where T, R and C are constants, respectively the table name, the row name and the column name. Associations of cells contents with cells references are represented by *ground atoms* of the form $T(R, C)$:N, where N is a constants. It should be noted that tables, rows and columns are "reified": they belong to the same domain as cell contents. This technique stems from class reification done in F-logic [15] and provides symmetric treatment to cell references and cell contents.

A *2-dimensional table* 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 A table named *school* describing students' results in mathematics and physics, together with their group membership can be represented by the following set of cells:
$\{school(kate, math)$:15, $school(kate, phys)$:12,
$school(kate, group)$:$gr1$, $school(mike, math)$:14,
$school(mike, phys)$:12, $school(mike, group)$:$gr2$,
$school(john, math)$:10, $school(john, group)$:$gr1$,
$school(alan, math)$:15, $school(alan, phys)$:16,
$school(alan, group)$:$gr2$, $school(suzy, math)$:12,
$school(suzy, phys)$:11, $school(suzy, group)$:$gr1\}$
This table can also be graphically depicted as shown in figure 1. □

school	*math*	*phys*	*group*
kate	15	12	gr1
mike	14	12	gr2
john	10		gr1
alan	15	16	gr2
suzy	12	11	gr1

Figure 1: The table school

2.2 Rule-Based Language

Rules à la Datalog are used to define new cell references and their associated contents from existing cells. We simply give their intuitive meaning. Consider the rule $p(X,Z) \longleftarrow q(X,Y), r(Z,Y)$. The standard (Datalog) informal meaning of this rule is that *if q(X,Y) holds and r(Z,Y) holds, then p(X,Z) 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(Z,Y), then there is a cell of reference p(X,Z).* We also add the handling of cell contents, and then a typical rule will be: $p(X,Z){:}W \longleftarrow q(X,Y){:}W, r(Z,Y){:}X$. 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(Z,Y) containing X, then there exists a cell of reference p(X,Z) containing W.*

A first example shows how rules can be used to specify table consolidation. In our examples, *is*, $\neq$, $+$ and $/$ are built-in predicates, having their standard meaning.

Example 2.2 Definition of a column in table *school* which gives the average for each student:

$$\begin{aligned} school(N, avg){:}Z \longleftarrow\ & school(N, math){:}X, \\ & school(N, phys){:}Y, \\ & Z \textit{ is } (X+Y)/2. \end{aligned}$$

A representation of the corresponding cells is depicted in Figure 2. □

school	*math*	*phys*	*avg*	*group*
kate	15	12	13.5	gr1
mike	14	12	13	gr2
john	10			gr1
alan	15	16	15.5	gr2
suzy	12	11	11.5	gr1

Figure 2: Computing averages

The rules can also be used to define various table restructurings. To this end, a higher-order syntax stemming from Hilog [6], allows variables to range over every constants used in cell references or cell contents. This provides for some kind of schema browsing abilities in the spirit of F-logic [15].

The following example illustrates how to use cell contents to build table names in order to restructure data in several tables.

Example 2.3 Splitting the table *school* into two tables according to the student groups:

$$\begin{aligned} G(N,M){:}X \longleftarrow\ & school(N,M){:}X, \\ & school(N,group){:}G, \\ & M \neq group. \end{aligned}$$

A representation of the corresponding tables are given Figure 3. □

*gr*1	*math*	*phys*	*avg*
kate	15	12	13.5
john	10		
suzy	12	11	11.5

*gr*2	*math*	*phys*	*avg*
mike	14	12	13
alan	15	16	15.5

Figure 3: A table for each group

3 Spreadsheet generation

In this section, we present a generation technique of a spreadsheet from its rule-based specifications. The main advantage of this technique is to suppress human encoding errors at a physical level. First we describe the global principle of the method. Next, we illustrate the spreadsheet generation on an example. This technique has been implemented in SWI-Prolog to produce Excel 5 [17] spreadsheets.

3.1 Principle

We distinguish between the spreadsheet development and its use. For us, the development stage includes every steps from the specification to the production of a spreadsheet. The use of a spreadsheet consists in (1) opening a spreadsheet with a spreadsheet software (e.g., Excel), (2) inputing data (characters, numbers) in some squares and (3) consulting the computed values obtained in other squares.

During the development stage, the spreadsheet is logically modeled by a rule-based program. The data available at this stage (called *initial inputs*). are used to generate the physical grid of the spreadsheet. Once opened with a spreadsheet software, this spreadsheet is ready to operate on user-supplied data (called *deferred inputs*).

At the targeted physical level in the current spreadsheet softwares, a spreadsheet simply a grid of squares, each of them can be empty, or contain a constant or a formula with references to other squares. (We do not take into account macro-commands generation, as this possibility is too much software-dependent.) So the generation technique must answer the following questions:

- What are the cell references to be implemented in the spreadsheet?
- Which cells contain a deferred input?
- What are the constants or the formulae contained in the other cells (not containing a deferred input)?

Starting from the rule-based specifications of a spreadsheet, the generation process we propose consists of four steps :

1. add rules specifying the cells used as deferred inputs;
2. using this new set of rules and the initial inputs, compute the set of useful references, i.e., references of cells that will contain an initial input, a formula or a deferred input;
3. specialize the set of rules obtained in step 1 with references computed in step 2, in order to display formulae which link cells;
4. implement the spreadsheet from the results of step 2 and step 3, using a binding schema that maps cell references to physical squares in the grid.

3.2 Generation technique

We consider the following example. Let the table *school* of example 2.1 be known and described by a set of ground atoms. Suppose we want to implement a spreadsheet to handle the marks of the students in group $gr2$. Suppose moreover we must take into account that these students will take one more course, computer science (*cs* for short), and can also take an optional course, *chemistry*. We want the spreadsheet to compute the final average of the students, when providing their marks in *cs* and *chemistry*. That is, we want to produce the spreadsheet depicted in figure 4.

gr2	*math*	*phys*	*cs*	*chemistry*	*finalAvg*
mike	14	12			
alan	15	16			

Figure 4: The spreadsheet to be generated

The final average $finalAvg$ is computed by applying coefficient 2 to the *math* and *phys* marks, and coefficient 1 to both *cs* and *chemistry*. If a student doesn't take *chemistry*, the string *nc* (for *non chosen*) is provided instead of a mark, and its final average is computed by applying coefficient 2 to the *math*, *phys* and *cs* marks. This can be specified by the following program P, consisting of three rules:

R_1

$$
\begin{aligned}
gr2(N, M){:}X \longleftarrow\ & school(N, M){:}X,\\
& school(N, group){:}gr2,\\
& M \neq group.
\end{aligned}
$$

R_2

$$
\begin{aligned}
gr2(N, finalAvg){:}A \longleftarrow\ & gr2(N, math){:}M1,\\
& gr2(N, phys){:}M2,\\
& gr2(N, cs){:}M3,\\
& gr2(N, chemistry){:}M4,\\
& number(M4)^1,\\
& A\ is\ (2 \times M1 + 2 \times M2 + M3 + M4)/6
\end{aligned}
$$

[1] $number(M4)$ succeeds if $M4$ is bound to a number.

R_3

$$\begin{aligned} gr2(N, finalAvg){:}A \longleftarrow\ & gr2(N, math){:}M1, \\ & gr2(N, phys){:}M2, \\ & gr2(N, cs){:}M3, \\ & gr2(N, chemistry){:}nc, \\ & A\ is\ (2 \times M1 + 2 \times M2 + 2 \times M3)/6 \end{aligned}$$

First step The developer must specify the reference of the cells that will contain a deferred input. In our example, these cells are those containing the marks in *cs* and *chemistry* in table $gr2$, while the cells containing the student groups and marks in *math* and *phys* in table *school* are the initial inputs. The developer specifies how to obtain the cell references of the deferred inputs, eventually by using cell references of the initial inputs. This specification is done by the following rules, in which a particular constant @ denotes the unknown contents of a deferred input:

RD_1

$$gr2(N, cs){:}@ \longleftarrow school(N, group){:}gr2.$$

RD_2

$$gr2(N, chemistry){:}@ \longleftarrow school(N, group){:}gr2.$$

Informally, RD_1 (resp. RD_2) says that if it exists a cell $school(N, group)$ containing $gr2$, then it exists a cell $gr2(N, cs)$ (resp. $gr2(N, chemistry)$) whose contents will be supplied later by the user of the spreadsheet.

Second step The semantics of the program P with respect to a set of initial inputs I, and a set of deferred inputs D is the minimal model[2] of P including both I and D [9]. This minimal model can not be computed at generation time since D will be supplied later by the end-user. However, it is already possible to compute an overestimation of this model. Let P' be the union of P and rules RD_1 and RD_2. To overestimate the minimal model of P (including both I and D), we use a naive bottom-up evaluation strategy to compute the minimal model of P' (including I) by considering @ as a constant like any other constants, except that:

1. if @ occurs in the right hand side of an *is* expression, then this right hand side reduces to @ (for example, $(A\ is\ (2 \times 14 + 2 \times 12 + @ + @)/6)$ gives $A\ is\ @$). For the sake of simplicity, we suppose that the left hand side of an *is* expression is always a free variable at evaluation time.
2. if @ occurs in a *test* expression (e.g., $@ + 1 > 4$, $number(@)$) then the test succeeds.
3. @ approximates any other constant (e.g., if $gr2(mike, chemistry){:}@$ holds then $gr2(mike, chemistry){:}nc$ also holds).

In the example, $\mathsf{I} = \{school(kate, math){:}15, school(kate, phys){:}12,$
$school(kate, group){:}gr1, school(mike, math){:}14,$

[2]For a definition of the minimal model of Datalog programs with respect to a set of ground atoms, see for example [2].

$school(mike, phys){:}12, school(mike, group){:}gr2,$
$school(john, math){:}10, school(john, group){:}gr1,$
$school(alan, math){:}15, school(alan, phys){:}16,$
$school(alan, group){:}gr2, school(suzy, math){:}12,$
$school(suzy, phys){:}11, school(suzy, group){:}gr1\}$

The minimal model of P' including I is
$\mathsf{A} = \mathsf{I} \cup \{gr2(mike, math){:}14, gr2(mike, phys){:}12,$
$gr2(alan, math){:}15, gr2(alan, phys){:}16,$
$gr2(mike, cs){:}@, gr2(mike, chemistry){:}@,$
$gr2(alan, cs){:}@, gr2(alan, chemistry){:}@,$
$gr2(mike, finalAvg){:}@, gr2(alan, finalAvg){:}@\}$

Considering that @ can approximate any other constant A is an overestimation of the minimal model of P for the initial inputs I and for all possible deferred inputs D specified in P'. This approximation is correct if:

1. P, I and D are consistent with each other, that is they don't give rise to inconsistent information. Inconsistency arise if incompatible cell contents are associated to the same cell reference. Automatic detection of inconsistency when D is unknown is beyond the scope of this paper and is considered in [16];
2. every cell references are fully determined by P' and I. This means that the cell references can not depend on D. Indeed, once the spreadsheet is generated, it is no more possible to automatically add new cells which references are computed from user-supplied data. This condition can be ensured automatically by testing that no constant @ occurs within a cell reference in A.

Third step In the third step, the program P is specialized with respect to A. This specialization aims at producing ground rules, for deriving the formulae to be implemented in the spreadsheet. Each rule of P is replaced with the set of its ground instances using atoms in A. As A overestimates the minimal model of P, this specialization is correct.

In order to preserve the coreferences during specialization, we differentiate explicitly the unknown contents @ of the cells in A. We simply associate a different constant $@_i, i \in \mathbb{N}$ with each different cell reference having an unknown contents.

The new set A becomes:
$\mathsf{A} = \mathsf{I} \cup \{gr2(mike, math){:}14, gr2(mike, phys){:}12,$
$gr2(alan, math){:}15, gr2(alan, phys){:}16,$
$gr2(mike, cs){:}@_1, gr2(mike, chemistry){:}@_2,$
$gr2(alan, cs){:}@_3, gr2(alan, chemistry){:}@_4,$
$gr2(mike, finalAvg){:}@_5, gr2(alan, finalAvg){:}@_6\}$

Remark A It should be noticed that the constants $@_i, i \in \mathbb{N}$ are only placeholders, and two constants $@_j$ and $@_k$ with $j \neq k$ can be associated with the same final constant when the deferred input is known. So during the instantiation process, we allow two occurrences of the same variable to be substituted by two different constants $@_j$ and $@_k$ and we preserve the coreference by adding a test ($@_j = @_k$) in the rule instance body. This aspect is not illustrated in the example presented here. The interested reader is referred to [16]. □

Let $\mathsf{P_{inst}}$ be P where $\mathsf{R_1}$, $\mathsf{R_2}$ and $\mathsf{R_3}$ are replaced by their ground instances. $\mathsf{P_{inst}}$ is:

R'$_1$

$$\begin{aligned} gr2(mike, math){:}14 \longleftarrow\ & school(mike, math){:}14,\\ & school(mike, group){:}gr2,\\ & math \neq group. \end{aligned}$$

R'$_2$

$$\begin{aligned} gr2(mike, phys){:}12 \longleftarrow\ & school(mike, phys){:}12,\\ & school(mike, group){:}gr2,\\ & phys \neq group. \end{aligned}$$

R'$_3$

$$\begin{aligned} gr2(alan, math){:}15 \longleftarrow\ & school(alan, math){:}15,\\ & school(alan, group){:}gr2,\\ & math \neq group. \end{aligned}$$

R'$_4$

$$\begin{aligned} gr2(alan, phys){:}16 \longleftarrow\ & school(alan, phys){:}16,\\ & school(alan, group){:}gr2,\\ & phys \neq group. \end{aligned}$$

R'$_5$

$$\begin{aligned} gr2(mike, finalAvg){:}@_5 \longleftarrow\ & gr2(mike, math){:}14,\\ & gr2(mike, phys){:}12,\\ & gr2(mike, cs){:}@_1,\\ & gr2(mike, chemistry){:}@_2,\\ & number(@_2),\\ & @_5\ is\ (2 \times 14 + 2 \times 12 + @_1 + @_2)/6 \end{aligned}$$

R'$_6$

$$\begin{aligned} gr2(alan, finalAvg){:}@_6 \longleftarrow\ & gr2(alan, math){:}15,\\ & gr2(alan, phys){:}16,\\ & gr2(alan, cs){:}@_3,\\ & gr2(alan, chemistry){:}@_4,\\ & number(@_4),\\ & @_6\ is\ (2 \times 15 + 2 \times 16 + @_3 + @_4)/6 \end{aligned}$$

R'$_7$

$$\begin{aligned} gr2(mike, finalAvg){:}@_5 \longleftarrow\ & gr2(mike, math){:}14,\\ & gr2(mike, phys){:}12,\\ & gr2(mike, cs){:}@_1,\\ & gr2(mike, chemistry){:}nc, \end{aligned}$$

$$@_5 \ is \ (2 \times 14 + 2 \times 12 + 2 \times @_1)/6$$

R'$_8$

$$\begin{aligned} gr2(alan, finalAvg){:}@_6 \longleftarrow\ & gr2(alan, math){:}15, \\ & gr2(alan, phys){:}16, \\ & gr2(alan, cs){:}@_3, \\ & gr2(alan, chemistry){:}nc, \\ & @_6 \ is \ (2 \times 15 + 2 \times 16 + 2 \times @_3)/6 \end{aligned}$$

The most interesting aspect of these rules is that they describe how to compute, from user-supplied values, the contents of cells intentionally defined from cells concerned with deferred inputs. For example, the rules R'$_5$ and R'$_7$ defines the contents of the cell reference $gr2(mike, finalAvg)$, and correspond to the following pseudo-code (not optimized):

```
IF $gr2(mike, cs)$ contains a number
  AND $gr2(mike, chemistry)$ contains a number THEN
  $gr2(mike, finalAvg)$ contains $((2\times14+2\times12+gr2(mike, cs)+gr2(mike, chemistry))/6)$
ELSE
  IF $gr2(mike, cs)$ contains a number
  AND $gr2(mike, phys)$ contains the string $nc$ THEN
  $gr2(mike, finalAvg)$ contains $((2 \times 14 + 2 \times 12 + 2 \times gr2(mike, cs))/6)$
  ELSE
  $gr2(mike, finalAvg)$ contains the string $undefined$
  ENDIF
ENDIF
```

The rules of P$_{\mathsf{inst}}$ are used in the next step to generate the formulae in the spreadsheet.

Fourth step The grid at the physical level is now filled with the appropriated contents. This is performed by the following operations:

1. A physical location in the grid is associated with each cell reference and the textual labels (i.e., indications row and column names) are placed.
2. The squares in the grid that contain a deferred input are filled with a particular symbol (e.g., a question mark).
3. The known constants (i.e., initial inputs or values derived from initial inputs) are placed in the grid.
4. And finally the formulae are generated in the appropriated squares.

The first operation can be done using a default binding of the cell references in the grid, if it is not specified by the developer (via textual constraints or interactively with a dedicated tool).

During the next operation, squares used for inputing data can be determined automatically, since they correspond to cell references associated with a contents which is not known and not defined by a rule. These cells are the cells associated with a constant @ in A, and such that their references do not appear in the head of a rule in P$_{\mathsf{inst}}$. In our example, these cells are those having the following references:

$\{gr2(mike, cs), gr2(mike, chemistery), gr2(alan, cs), gr2(alan, chemistry)\}$

Concerning the third operation, the set of cells containing known constants (when D is unknown), corresponds to the minimal model of P containing I. In the example, this set is:

$\mathsf{I} \cup \{gr2(mike, math) : 14, gr2(mike, phys) : 12,$
$\quad gr2(alan, math) : 15, gr2(alan, phys) : 16\}$

and using the physical binding, the corresponding squares can be filled.

Finally, for the last operation, the set of squares containing a formula corresponds to the set of cell references in A that are not associated with deferred inputs or known constants (determined during operations 2 and 3). In our example, the references of cells containing a formula are $gr2(mike, finalAvg)$ and $gr2(alan, finalAvg)$.

For each such cell reference, we consider the rules in $\mathsf{P}_{\mathsf{inst}}$ defining its contents (i.e., the set of rules in $\mathsf{P}_{\mathsf{inst}}$ such that this cell reference appears in the rule head). As mentioned in the third step, this set of rules corresponds to a pseudo-code that can be used to compute the cell contents. At the physical level, the conditionals set up in the pseudo-code are encoded in formulae using the IF function. This function uses three parameters: $IF(Exp1; Exp2; Exp3)$, where $Exp1$ is a boolean expression and where both $Exp2$ and $Exp3$ are (unrestricted) expressions. If $Exp1$ evaluates to $true$ then $IF(Exp1; Exp2; Exp3)$ evaluates to $Exp2$, otherwise it evaluates to $Exp3$.

We consider the physical binding corresponding to the spreadsheet depicted in figure 4. So, for example, the cell reference $gr2(mike, cs)$ is associated in the grid to square $L2C4$ (line 2, column 4), while $gr2(mike, chemistry)$ and $gr2(mike, finalAvg)$ are associated respectively to squares $L2C5$ and $L2C6$. And then, the rules defining the contents of $gr2(mike, finalAvg)$ (corresponding to the pseudo-code given in the third step) are encoded in square $L2C6$ by the following formula (using Excel 5 syntax [17]):

$= IF(AND(ISNUM(L2C4); ISNUM(L2C5));$
$\quad (2 \times 14 + 2 \times 12 + L2C4 + L2C5)/6;$
$\quad IF(AND(ISNUM(L2C4); L2C5 = "nc"); (2{\times}14{+}2{\times}12{+}2{\times}L2C4)/6; "undefined"))$

where AND is the boolean conjunction operator and $ISNUM(X)$ evaluates to $true$ if and only if X evaluates to a number.

4 Conclusion

Experimental studies [5, 8, 18] have pointed out that spreadsheets contain a lot of errors, and since spreadsheets are widely used to support human decisions these errors have an increasing economical impact.

To limitate the possible sources of errors we propose to specify spreadsheets at a logical level, and to generate automatically their physical counterparts. The logical level is described by using a rule-based language dedicated to multidimensional databases [9, 11] and the targeted physical level is a grid containing constants and formulae. Other aspects of the relationship between this kind of language and the life-cycle of spreadsheets are studied in [3, 4]. The technique presented in the paper is used in a generator written in SWI-Prolog [19] to produce Excel 5 [17] spreadsheets from their logical definitions. To our knowledge this is the first tool allowing such an automatic process. Its two main benefits are : (1) clear and easily maintainable specifications can be written at an abstract level, and (2) the automatic generation of the physical level suppresses human encoding errors.

Future extension of our work includes the handling of aggregates in the spreadsheet generation and the support of data exchange between spreadsheets and relational databases.

References

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

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

[3] 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.

[4] J.-F. Boulicaut, P. Marcel, and F. Pinet. Les tableurs : un talon d'achille pour les systèmes d'informations intégrés. Research report RR-98-02, LISI, INSA de Lyon, Jan. 1998. 28 pages, in french, submitted.

[5] P. Brown and P. Gould. An experimental study of people creating spreadsheets. *ACM TOIS*, 5(3):258–272, Jul. 1987.

[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] P. B. Cragg and M. King. Spreadsheet modeling abuse: an opportunity for or? *J. Opr. Res. Soc.*, 1992.

[9] M. S. Hacid, P. Marcel, and C. Rigotti. A rule-based CQL for 2-dimensional tables. In *Proc. 2nd Int. Workshop on Constraint Database Systems (CDB'97)*, volume 1191 of *LNCS*, pages 92–104, Delphi, Greece, Jan. 1997.

[10] M. S. Hacid, P. Marcel, and C. Rigotti. A rule-based data manipulation language for OLAP systems (short paper). In *Proc. 5th DOOD*, volume 1341 of *LNCS*, pages 417–418, Montreux, Switzerland, Dec. 1997.

[11] M. S. Hacid, P. Marcel, and C. Rigotti. A rule-based language for ordered multidimensional databases. In *Proc. 5th Int. Workshop on Deductive Database and Logic Programming (DDLP'97)*, volume 317 of *GMD-Studien*, pages 69–81, Leuven, Belgium, Jul. 1997.

[12] M. S. Hacid, P. Marcel, and C. Rigotti. Rule-based restructuring and summarizing in multidimensional databases. In *Actes des Journées Francophones de Programmation en Logique et par Contrainte*, Nantes, France, May. 1998. Hermès.

[13] J.-L. Hainaut. *Bases de données et modèles de calcul : outils et méthodes pour l'utilisateur.* InterEditions, 1994.

[14] T. Isakowitz, S. Schocken, and H. C. Lucas JR. Towards a logical/physical theory of spreadsheet modeling. *ACM TOIS*, 13(1):1–37, Jan. 1995.

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

[16] P. Marcel. *Manipulations de données multidimensionnelles et langages de règles.* PhD thesis, INSA de Lyon, 1998. in preparation.

[17] Microsoft. *EXCEL V5.0 user's guide*, 1993. 871 p.

[18] R. R. Panko and R. P. Halverson. Spreadsheets on trial: a survey of research on spreadsheet risks. In *Proceedings of the Twenty-Ninth Hawaï International Con-ference on System Sciences*, Maui, Hawaï, Jan. 1996.

[19] J. Wielemaker. *SWI-Prolog 2.9.6.* Dept. of Social Science Informatics (SWI). University of Amsterdam, Roeterstraat 15, 1018 WB Amsterdam, The Netherlands, 1997.