Automatic Aggregation using Explicit Metadata
(Extended Abstract)

Stéphane Grumbach
INRIA
Rocquencourt BP 105
78153 Le Chesnay, France
Stephane.Grumbach@inria.fr

Leonardo Tininini
CNR-IASI
viale Manzoni 30
00185 Roma, Italy
tininini@iasi.rm.cnr.it

Abstract

The paper presents a logical data model for statistical data with an explicit modelization of metadata, which allows to perform automatic aggregation. The data are stored in standard relations from the relational model, while the metadata, defining the semantics of the relations, are represented by a sort of functional dependencies for summary attributes, called numerical dependencies, which specify the function defining the summary values in terms of micro-data, as well as the interrelationships among summary values. Relations with numerical dependencies can be considered as statistical views over initial relations over micro-data which are generally not accessible. The main benefit of the present model is to support declarative queries, that do not need to bother with the intricacy of metadata, which can be handled in a completely automatic manner. Queries can be asked against the views as well as the initial relations, and in this later case answered using the views. The numerical dependencies are run together with the query to compute (when possible) the answer. The mechanism is illustrated by an implementation in Prolog of various examples of queries and dependencies. It is shown in particular that various classical problems in the realm of statistical and multidimensional databases can be easily modeled and solved in the present framework. Finally, the proposed formalism is shown to be useful for statistical database schema design.

1. Introduction

The manipulation of aggregate data raises important and specific issues, that have been studied with different foci both in the field of statistical databases (SDBs) [Sho82, Su83, SW85, Gho86, OOM87, CDL95, RBT96], and in the field of on-line analytical processing (OLAP) [GBLP96, HRU96, LS97, Sho97, CD97]. The real challenge of this sort of data is caused by the rather intricate semantics of summary values, that is not handled by classical database systems. The relationships between OLAP and data mining have also been investigated by several authors [GMPS96, Han97, Han98].

The research on statistical databases has been mainly concerned with conceptual modelization. The focus is on the *macro-data* obtained by grouping and aggregating the original *micro-data*. In SDBs, it is generally assumed that the micro-data are not available, both for efficiency reasons (the micro-data are too large, not relevant, and direct access to summarized data is desirable), and for privacy reasons (personal information should not be public). The semantics of macro-data is generally explained in terms of *meta-data* [McC82], a concept that suffers from fuzziness in many approaches.

In the present paper, we propose a representation of statistical data in terms of relations of the relational model, whose semantics is given by a new kind of functional dependencies, called numerical dependencies. In addition to functional dependencies, the numerical dependencies specify explicitly the definition of the function. This makes full sense in the context of interpreted numeric domains. For example, a relation containing an attribute which is the average of a collection of values, (e.g. average age of people in companies) is associated to a numerical dependency (e.g. company → avg-age) which specifies the means to compute the average age from the ages of persons.

There has been two main directions in the modeling of aggregate data both in OLAP systems and in SDBs, the former based on the relational model (cf ROLAP), and the latter based on ad-hoc structures and operators, specifically introduced to model such data (cf MOLAP and statistical graphical models).

In the relational approach two sorts of attributes are often distinguished [CS81, RR83, OOM85, Raf90]: (i) *summary attributes* representing the result of the application of an aggregation function on micro-data (their numerical values being called summary data); and (ii) *category attributes*, which provide a qualitative description of the summary attributes and are generally assumed to be part of the metadata. The category attributes are key-attributes and a tuple is stored for each summary value. There is a large variety of such models, among which some include integrity constraints to be satisfied by relations representing statistical objects [CS81, RR83, Su83, Gho86]. The main weakness of these models is that a significant portion of information is not formally specified and remain hidden in the textual description of the statistical objects, hence making difficult an effective combination of the information they contain.

In the ad-hoc approach, new graphical conceptual models [CS81, RR83, Su83, SR91, RBT96] based on original data structures with new operators are proposed. The advantage is a better representation of the semantics of data. Nevertheless, the manipulation of data depends in a crucial way on the correct usage of the defined operators and requires the intervention of an "expert" user.

In the model we propose, the statistical data are stored simply in relations. We consider a language for statistical properties, based on real arithmetic and group-by operations, to perform count, sum and product of collections of values [GRT99]. The meaning of the relations are captured through numerical dependencies, expressed in terms of Horn clauses in the language for statistical properties.

Storing statistical data by conventional relations has some important consequences. Statistical DBs can be seen as a collection of precomputed aggregate views over a database of micro-data. The queries against the SDB can be expressed by referring either to the precomputed aggregate views or to the micro-data relations (by specifying the required aggregation). Since the micro-data are assumed to be unavailable, the problem is to transform the given query into an equivalent one referring only to the precomputed views. This problem is widely known in literature as the *view usability problem* and it is of great relevance in several fields of database research.

In the present context, it is shown that if the metadata describing the relationships among micro- and macro-data have been stated using numerical dependencies, the user has only to express in a declarative manner the required information and the level of detail, without worrying about the data that are really materialized in the database. The system will produce the necessary rewriting, based on the available precomputed views and on the stored dependencies.

The mechanism is illustrated by an implementation in Prolog of various examples of queries and dependencies. It is shown that various classical problems in the realm of statistical and multidimensional databases can be easily modeled and solved in the present framework, particularly a challenging problem proposed by Lenz and Shoshani in [LS97].

The paper is organized as follows. In the following section, we discuss examples of statistical tables that illustrate various aspects of the complexity of the data. In Section 3, we present the language to express statistical properties. Section 4 is devoted to the new data model based on relations with numerical dependencies. In Section 5, we illustrate how the model can be used to query statistical data in a declarative fashion.

2. Statistical data

In this section, we present examples of statistical data and describe the complex semantical dependencies between the data. One of the most common use of statistical data is in demographic studies, for instance in the analysis of population distribution. We first present such an example.

Example 1 Consider the table of Figure 1, displaying the population (in 1996) of some European countries classified by sex. Such a table can be computed by selecting the appropriate census micro-data (relative to 1996), and grouping them by attributes Sex and Country (only by Country for the Total values) and finally by performing a count operation on such groups.

Country	Fpop	Mpop	Total
France	28.299	29.810	58.109
Italy	27.926	29.534	57.460
United Kingdom	28.215	30.080	58.295
Spain	19.347	20.057	39.404

Figure 1. Population in 1996 (in millions)

Another typical example of demographic data is shown in Figure 2, where the population density values (number of inhabitants per square kilometer) of some countries are given. In this case, a different aggregation function is applied which is obtained by the ratio of other macro data, namely the area of the countries and their population in the respective years.

Country	1960	1970	1980	1990
France	84	93	99	104
Italy	167	177	187	189
United Kingdom	215	228	231	235
Spain	60	67	74	77

Figure 2. Population density in European countries

A fundamental operation in the context of statistical database is the *summarization* of one or more category attributes, i.e. the operation which allows for the "removal" of one or more classifying attributes or the aggregation at a lower level of detail with the consequent recomputation of the summary values. In the *On Line Analytical Processing* (OLAP) terminology, this is known as *rolling up* over one dimension. For example, the female and male populations can be summed to obtain the total population or the population values of the countries can be summed to obtain the population values of the European Community, provided that the composition of the European Community is known (the composition through the years could be represented by the relation of Figure 3 listing the dates of entry of countries in the EC)

Country	Membership year
France	1958
Italy	1958
United Kingdom	1973
Spain	1986
...	...

Figure 3. Membership to the European Community

Whether a summarization can be performed or not without accessing the original micro-data, is a fundamental question, which strictly depends on the type of aggregation performed over the micro-data. It is obvious for instance that the Total values of Figure 1 representing the summarization (roll-up) of the Sex attribute can be obtained by summing the female and male corresponding values, while the population density of the EC cannot be obtained by the densities of its member states. □

The next example refers to stock market variations and shows that rolling up over one dimension may require an operation different from addition to be performed, unlike the foregoing example on population values.

Example 2 The table of Figure 4 presents the variation index of some shares in the indicated periods. The variation index is defined as the ratio between the final and the initial value of the share:

$$I_{var}(share_i) = \frac{val_{in}(share_i)}{val_{fin}(share_i)}$$

Since the final value of each share in a given month coincides with the initial value of the same share in the following month, the index relative to the year can be obtained by multiplying all the corresponding month values.

The next example was first given in [LS97]. It is particularly interesting for its complex semantic constraints that

Shares	Jan	Feb	...	Dec	1996
ACME	1.08	1.05	...	0.98	1.23
Toon Ltd.	1.01	0.98	...	1.00	1.02
...	...	...	...	...	...

Figure 4. Variation indexes of the stock market

are often embedded in statistical aggregate data and represents somehow an open challenge in the modeling of statistical and multidimensional databases.

Example 3 The table of Figure 5 presents the number of students registered in two departments, assuming that each student attended exactly a 2-year program and that no student started his/her program before 1994. Consequently, the total column values cannot simply be obtained by adding the figures for each year.

Department	1994	1995	1996	Total
CS	15	17	13	28
statistics	10	15	11	21
Total	25	32	24	49

Figure 5. Number of students by department and year

Nevertheless, the total values can be correctly obtained from the existing data. For example, in the computer science department, there were 15 first-year students in 1994, then 15 second-year and 2 first-year students in 1995 and finally 2 second-year and 11 first-year students in 1996. The total number of students that attended over the 1994-1996 period was therefore $15 + 2 + 11 = 28$, as displayed in the Total column. □

In the rest of the paper, we will show that the summarization of the attribute Year in the table of Figure 5 can be automatically obtained, using numerical dependencies. The user therefore does not need to write a complex query to recover the right number hidden in the table. It suffices to express declaratively what he wants to count and the system solves the how.

3. Aggregate Data Language

The language used to define aggregate data in this paper is based on an extension of first order logic over the structure of the real field, as proposed in [GRT99]. We review here the main definitions and concepts.

The mathematical context is the structure of the real field $\mathcal{R} = (\mathbb{R}, \leqslant, +, \times, 0, 1)$ with universe $\mathbb{R}$, the set of reals, over the signature consisting of predicates, = and $\leqslant$, functions $+$ and $\times$, and two constants 0 and 1. We denote by $\text{FO}_{\mathbb{R}}$ first-order logic over this signature. The signature is extended to aggregation (second-order) operators defined as follows. Let $\varphi(x, \bar{y})$ be a formula with free variables $x, \bar{y}$, and $\psi(x, \bar{y}, z)$ with free variables $x, \bar{y}, z$.

- **count** $C_x\varphi(x, \bar{y})$ defines a partial function over attributes $\bar{y}$, whose domain is $\{\bar{y} | \exists x \varphi(x, \bar{y})\}$ and defined as follows: a tuple $\bar{a}$ is mapped to the cardinality $c_{\bar{a}}$ of the set $\{x | \varphi(x, \bar{a})\}$.
- **sum** $\Sigma_x^z \psi(x, \bar{y}, z)$ defines a partial function over attributes $\bar{y}$, whose domain is $\{\bar{y} | \exists x, z \psi(x, \bar{y}, z)\}$ and defined as follows: a tuple $\bar{a}$ is mapped to the sum $s_{\bar{a}}$ of the elements of the multiset $\{\!\{x | \psi(x, \bar{a}, z)\}\!\}$.
- **multiply** $\Pi_x^z \psi(x, \bar{y}, z)$ defines a partial function over attributes $\bar{y}$, whose domain is $\{\bar{y} | \exists x, z \psi(x, \bar{y}, z)\}$ and defined as follows: a tuple $\bar{a}$ is mapped to the product $p_{\bar{a}}$ of the elements of the multiset $\{\!\{x | \psi(x, \bar{a}, z)\}\!\}$.

The operators *sum*, and *multiply* can also be used on sets of values versus multisets, with no *multiset argument* z, and we write $\Sigma_x \varphi(x, \bar{y})$ and $\Pi_x \varphi(x, \bar{y})$.

For readability, we use the overlined notation

$$\overline{C_x \varphi(x, \bar{y})}(\bar{y})$$

to distinguish between the variables of the formula φ and the arguments of the function. The *arguments* of the function correspond to the free variables of the formula which are not bound by the aggregation.

We consider the first-order logic $\text{FO}_{\mathbb{R}}^{agg}$ over the signature of the real field with the three aggregation operators introduced above. The definition of terms and formulae of the language are then standard. Note that terms and formulae admit mutually recursive definitions (for further details on the language $\text{FO}_{\mathbb{R}}^{agg}$ and on its properties, see [GRT99].

The following example illustrates the definitions, particularly the role of the multiset argument in the aggregation operators defined above.

Example 4 Let us consider an atomic formula $R(x, y, z)$ instantiated as follows:

R

x	y	z
1	a	c
2	a	c
2	a	d
1	b	c
2	b	d

The function $\overline{\Sigma_x R(x,y,z)}(y, z)$ depends upon variables y and z and has the following non-zero values.

$s = \overline{\Sigma_x R(x,y,z)}(y, z)$

y	z	s
a	c	3
a	d	2
b	c	1
b	d	2

The function $\overline{\Sigma_x^z R(x,y,z)}(y)$ depends instead upon variable y and has the following non-zero values. Note the role of the multiset parameter for $y = a$, which forces to sum twice the value $x = 2$.

$s = \overline{\Sigma_x^z R(x,y,z)}(y)$

y	s
a	5
b	3

□

Note that the role of multiset parameters in $\text{FO}_{\mathbb{R}}^{agg}$ is essentially dual to that of **group by** clauses in SQL. For example the SQL query:

select y, **sum**(x)
from R (x, y, z)
group by y

is expressed by the following $\text{FO}_{\mathbb{R}}^{agg}$ query:

$$\{[y, s] | s = \overline{\Sigma_x^z (R(x,y,z))}(y)\}$$

The **count-distinct** can also be easily defined from the count. For example the SQL query:

select y, **count**(**distinct** x)
from R (x, y, z)
group by y

is expressed by the following $\text{FO}_{\mathbb{R}}^{agg}$ query:

$$\{[y, s] | s = \overline{C_x(\exists z R(x,y,z))}(y)\}$$

Note that the SQL **distinct** clause (corresponding to an existential quantification in $\text{FO}_{\mathbb{R}}^{agg}$) destroys the multiplicities of the values and this fact has serious consequences on the possibility to perform summarizations. Particularly, the complex semantics of the table in Example 3 is mainly due to the fact that it has been obtained by a count-distinct aggregation over the micro-data.

Other functions can be expressed in the language, e.g. min, max, avg, percentage and also more complex statistical functions, like median and index values (see [GRT99] for details).

4. Statistical data model

In the previous section, we have considered the language $\mathrm{FO}_{\mathbb{R}}^{agg}$ to define aggregate functions, and express statistical aggregate queries. We now propose the data model based on simple relations whose attribute semantics is given by numerical dependencies expressed in the logic presented above.

The core concept of the model is that of *numerical dependency*, which expresses in formal manner the semantics of the aggregate attributes and the existing dependencies among micro- and macro-data.

Definition 1 A *numerical dependency* is a Horn clause of the form:

$$\forall x_1 \ldots \forall x_n R_1(t_1^1, \cdots, t_{n_1}^1) \wedge \cdots \wedge R_k(t_1^k, \cdots, t_{n_k}^k) \\ \Rightarrow x_i = F(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$$

where (i) all variables are universally quantified; (ii) all variables in the head appear in the body; (iii) all terms t_j^i in the body are variables among $x_1 \ldots x_n$; and (iv) $F(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n)$ is a term of numerical sort with no occurrence of the variable x_i. In the following, for brevity, we omit the universal quantifications.

Consider the relation R over signature `[Country, Sex, Population]` corresponding to the table of Figure 1. The following numerical dependency represents the property that the total population is equal to the sum of the female and male ones:

$$(c, female, p_f) \wedge R(c, male, p_m) \wedge R(c, all, p) \\ \Rightarrow p = p_f + p_m$$

Classification hierarchies can also be represented by numerical dependencies as shown in the next examples. Consider a relation R over signature `[Country, Pop]` and consider for instance the classification hierarchy: Benelux = { Belgium, Nederland, Luxembourg }. Then the population of Benelux should satisfy the following dependency:

$$R(Benelux, p) \wedge R(Belgium, p_1) \wedge R(Nederland, p_2) \\ R(Luxembourg, p_3) \Rightarrow p = p_1 + p_2 + p_3$$

In the more complex case described in Example 1, where the classification hierarchy from countries to EC varies over time, a slightly more complicated dependency is required (we assume that relation R is over the signature `[Country, Year, Population]` and relation S over the signature `[Country, MembershipYear]`):

$$R(EC, y, t) \Rightarrow t = \overline{\Sigma_p^{cm}(R(c,y,p) \wedge S(c,m) \wedge y \geqslant m)}(y)$$

We can now introduce our data model.

Definition 2 A *database schema* is specified by a structure $\tau = < \mathcal{A}, \sigma, \mathcal{F}, \mathcal{N} >$, where

- $\mathcal{A} = (A_1 : D_1, \cdots, A_n : D_n)$, is a collection of attribute names with their associated domains
- σ is a relational signature;
- $\mathcal{F}$ is a collection of functional dependencies; and
- $\mathcal{N}$ is a collection of numerical dependencies.

A *database instance* of schema $\tau = < \mathcal{A}, \sigma, \mathcal{F}, \mathcal{N} >$ is a relational instance of signature σ. Therefore a database instance is simply a classical relational database. Let us illustrate the previous definition with some examples.

Example 5 Consider again the census data of Example 1. The relational signatures are R`[Country, Year, Sex, Pop]`, S`[Country, Year, Area]` and T`[Country, Year, Density]`. The domains of the attributes are the following: $D_{\texttt{Country}} = \{France, Germany, Italy, ...\}$, $D_{\texttt{Sex}} = \{female, male\}$, $D_{\texttt{Year}} = \{1980, ..., 1996\}$, and the other domains are numerical $D_{\texttt{Area}} = D_{\texttt{Pop}} = D_{\texttt{Density}} = \mathbb{R}$. The special value "all" [GBLP96] is also included in the domain of Sex to represent the union of all values in the domain.

The functional dependencies are:

$$R.Country, R.Year, R.Sex \rightarrow R.Pop$$

$$S.Country, S.Year \rightarrow S.Area$$

$$T.Country, T.Year \rightarrow T.Density$$

Finally, the numerical dependencies are given by:

$$R(c, y, all, p) \wedge R(c, y, female, p_f) \wedge R(c, y, male, p_m) \\ \Rightarrow p = p_f + p_m$$

and

$$R(c, y, all, p) \wedge S(c, y, a) \wedge T(c, y, d) \Rightarrow d = \frac{p}{a}$$

□

Example 6 Consider again the market index of Example 2. The relational signature is R`[Share, Month, Year, VarIndex]`. The domains of the attributes are the following: $D_{\texttt{Share}} = \{ACME, ToonLtd, ...\}$, $D_{\texttt{Month}} = \{jan, ..., dec\}$, $D_{\texttt{Year}} = \{1980, ..., 1996\}$, $D_{\texttt{VarIndex}} = \mathbb{R}$. The special value "all" is also included in the domain of Month to represent the union of all values in the domain.

The functional dependency is:

$$Share, Month, Year \rightarrow VarIndex$$

Finally, the numerical dependency representing the roll-up over the time dimension is given by:

$$R(s, all, y, v_a) \Rightarrow v_a = \overline{\Pi_v^m(R(s,m,y,v))}(s, y)$$

□

Consider now the relation of Example 3.

Example 7 Let us examine at first the possible structure of micro-data from which the given aggregate data have been computed. Consider a relation R of micro-data over signature `[Dept, Year, Student]`. The macro-data of Figure 5 can be obtained in a relation R' over signature `[Dept, Year, #students]`, satisfying the numerical dependency:

$$R'(d, y, c) \Rightarrow c = \overline{C^x(R(d,y,x))}(d, y)$$

When computing the total number of students over the indicated years a duplicate elimination is required, because each student has to be counted only once. In practice a count-distinct has to be performed on relation R and this causes the impossibility of computing the total number by summing the values in each single year. The corresponding numerical dependency is given by:

$$R'(d, all, c) \Rightarrow c = \overline{C^x(\exists y R(d,y,x))}(d)$$

Let us express now formally the additional knowledge we have on the data. Particularly, we know that every student attended exactly a 2-year program and that no student started his program before 1994. We define a relation R'' over signature `[Dept, Year, #students]` representing the number of students attending the department d in the year y for *the first year*. The dependency:

$$R'(d, 1994, c) \wedge R''(d, 1994, c') \Rightarrow c' = c$$

expresses that no student started his program before 1994 (in 1994 all students are first year students), while the dependency:

$$R'(d, y, c) \wedge R''(d, y, c') \wedge R''(d, y-1, c'') \Rightarrow c' = c - c''$$

expresses that every student attended exactly a 2-year program (for each year the total number of students is the sum of the first year ones + the second year ones, which in turn are the first year students of the previous year).

Finally, since a count-distinct has to be performed when computing the total number of students over the years, we count each student only when attending the department for the first year (in this way each student is counted exactly once). This fact is expressed by the dependency:

$$R'(d, all, c) \Rightarrow c = \overline{\Sigma_c^y(R''(d,y,c) \wedge y \neq all)}(d)$$

□

5. Computing with numerical dependencies

The form of the numerical dependencies (Horn clauses) suggests their translation into the rules of a logic programming language like Prolog. In this section some of the foregoing examples are modeled by translating the numerical dependencies into Prolog rules and the computation of some queries is accomplished by using the Prolog inference engine.

The development of an inference mechanism specifically designed for statistical data management would be preferable, both in terms of ease of use and in terms of computational efficiency. However, the choice of a multi-purpose language is useful to show (i) the flexibility of the proposed approach, (ii) the minimum effort required to translate numerical dependencies into a different formalism, and (iii) the modeling power and increased answering capabilities made possible by the proposed approach.

Example 8 We first consider a demographic modeling based on Example 1. We consider - for limitations of space - a simplified composition of the European Community as shown in Figure 3. It is therefore assumed that the EC was constituted by France and Italy from 1958, and that UK and Spain joined the Community in 1973 and 1986, respectively. Note that the metadata part would not require any modification in a true database (where all EC states are considered). Only a higher number of "facts" (the purely relational part of the DB) would have to be considered.

The relation representing the member date of entry is modeled by using the following Prolog rules:

```
memberEC(italy,1958).
memberEC(france,1958).
memberEC(united_kingdom,1973).
memberEC(spain,1986).
```

We also assume the following population values are stored in the database (note that the population data of UK are only available in 1991 and classified by sex):

```
popdb(italy,1971,48).
popdb(france,1971,48).
popdb(spain,1971,45).
popdb(italy,1981,54).
popdb(france,1981,52).
popdb(spain,1981,50).
popdb(italy,1991,55).
popdb(france,1991,53).
popdb(spain,1991,52).
popBySexdb(united_kingdom,1991,f,27).
popBySexdb(united_kingdom,1991,m,25).
```

The annotated set of Prolog rules representing the metadata and used to emulate numerical dependencies and aggregate operators are the following:

```
/* the population value of a country
in a given year can be computed either
if it is in the DB or if the female and
male values are available */

 population(C,Y,Value) :-
            popdb(C,Y,Value).
 population(C,Y,Value) :-
            popBySexdb(C,Y,f,VF),
            popBySexdb(C,Y,m,VM),
            Value is VF+VM.

/* a country is member of the EC in a
given year if its member entry year is
less or equal to the given one */

 ismemberECin(Country,Year) :-
            memberEC(Country,Year1),
            Year1 =< Year.

/* List is the set of states member of
the EC in the given year */

 memberECstates(Year,List) :-
       setof(Country,
       isMemberECin(Country,Year),List).

/* corresponds to (a specialized ver-
sion of) the sum aggregate operator */

 sumPopInList([],0,Y).
 sumPopInList([H|T],S,Y) :-
            sumPopInList(T,S1,Y),
            population(H,Y,Value),
            S is S1+Value.

/* the population of the EC in a given
year is the sum of the population of
its member in the same year */

 popEC(Year,Number) :-
       memberECstates(Year,List),
       sumPopInList(List,Number,Year).
```

Finally, the database can be queried by simply using the "views" population and popEC, particularly:

```
?- popEC(1971,N).
N = 96
yes
```

computes the population of the EC in 1971, by using the available population values of Italy and France in that year.

```
?- popEC(1981,N).
no
```

the computation (correctly) fails because the population value of the UK in 1981 is not available

```
?- popEC(1991,N).
N = 212
yes
```

computes the population of the EC in 1991, by automatically aggregating the UK female and male values of population in that year. □

Example 9 We consider the statistical table of Example 3 summarizing the number of students registered in various departments. We denote by:

- $r1(D, Y, N)$ the predicate (relation) representing the global number N of students attending department D in the year Y.
- $r2(D, Y, N)$ the predicate (relation) representing the number N of students attending department D in the year Y for the first year.

The special value 'all' is used to represent the total values as in [GBLP96]. The stored part of $r1$ is represented by the following set of rules:

```
r1db(cs,1994,15).
r1db(stat,1994,10).
r1db(cs,1995,17).
r1db(stat,1995,15).
r1db(cs,1996,13).
r1db(stat,1996,11).
```

The annotated set of Prolog rules representing the metadata and used to emulate numerical dependencies and aggregate operators are the following:

```
/* these rules correspond to the
aggregate/group-by operators */
```

```
listsum([],0).
listsum([H|T],S) :- listsum(T,S1),
                    S is S1+H.
listcount([],0).
listcount([H|T],C) :- listsum(T,C1),
                      C is C1+1.


/* the summarization over the Year di-
mension is through r2 the summarization
over the Dep dimension is done directly
through r1 (see Example 7) */


summarizeYD(SN) :-
         bagof(N,D^r1(D,all,N),S1),
         listsum(S1,SN).
summarizeY(D,SN) :-
         bagof(N,Y^r2(D,Y,N),S1),
         listsum(S1,SN).
summarizeD(Y,SN) :-
         bagof(N,D^r1db(D,Y,N),S1),
         listsum(S1,SN).


/* some views to facilitate querying */


r1(D,Y,N) :- Y == all,
             D == all,
             !,
             summarizeYD(N).
r1(D,Y,N) :- Y == all,
             !,
             summarizeY(D,N).
r1(D,Y,N) :- D == all,
             !,
             summarizeD(Y,N).
r1(D,Y,N) :- r1db(D,Y,N).


/* no student attended before 1994 (all
students of 1994 are first year stu-
dents) */


r2(D,1994,N) :- r1db(D,1994,N).


/* the number of students attending a
department in a given year is the sum
of the number of first year students
and second year students /*

r2(D,Y,N) :- r1db(D,Y,N1),
             Y1 is Y-1,
             r2(D,Y1,N2),
             N is N1-N2.
```

Finally, the database can be queried by simply using the view $r1$ and the special value "all", particularly:

```
?- r1(cs,all,N).
N = 28
yes
```

computes the global number of students attending the department of computer science from 1994 to 1996, while the queries:

```
?- setof((D,N),r1(D,all,N),Answer).
D = _2667
N = _2668
Answer = [(cs,28),(stat,21)]
yes

?- setof((Y,N),r1(all,Y,N),Answer).
Y = _3685
N = _3686
Answer = [(1994,25),(1995,32),(1996,24)]
yes

?- setof((N),r1(all,all,N),Answer).
N = _4178
Answer = [49]
yes
```

computes the marginal values (Total column and row) of the table. □

6. Conclusion

The main contribution of the paper is the introduction of a formalism which allows to express queries over statistical data in a simple declarative way even when they require a complex computation involving metadata to be answered. The system performs the automatic aggregation of statistical information. This solves some open problems in the SDB community, particularly the problems related to:

- the summarization over a classification hierarchy that varies over time and not necessarily complete;
- the summarization over aggregate data that have been obtained by aggregate functions that loose multiplicity (like count-distinct);

- the combination of statistical data based on different aggregate functions and/or different classifications to obtain new statistical data (e.g. an average from a sum and a count having partially different classifying attributes);

The problem addressed in this paper can be seen as a particular case of the general problem of answering a (relational) query using some precomputed views, referred to as the *view usability problem*.

Two important issues should be addressed in this context. First, whether a query can be answered with the available data. This is not always the case as seen by some examples in the paper. It is in general undecidable if the statistical views contain enough information to do so. Nevertheless, we have shown in some simple cases that it could be decided. Second, if the query can be answered, what is the complexity of computing the answer using the statistical views available.

In particular, we have shown that if the query and the views are simple count or sums over a given relation (like in many practical OLAP situations), the rewriting and the computation can be obtained in linear time and the rewriting method is complete. On the other hand, the inclusion of ratio functions, like averages and percentages, increases the complexity to at least quadratic.

The theoretical investigation that is made possible by the explicit formalization of metadata suggest principles for statistical database schema design. In particular, it allows to characterize the amount of information lost in statistical views, and the cost of recovering information that has not been lost.

Acknowledgments
The authors wish to warmly thanks Maurizio Rafanelli for numerous discussions on this topic.

References

[CD97] S. Chaudhuri and U. Dayal. An overview of data warehousing and olap technology. *SIGMOD Record*, 26(1):65–74, 1997.

[CDL95] T. Catarci, G. D'Angiolini, and M. Lenzerini. Conceptual language for statistical data modeling. *Data and Knowledge Engineering*, 17(2):93–125, 1995.

[CS81] P. Chan and A. Shoshani. Subject: A directory driven system for organizing and accessing large statistical databases. In *Proceed. of the 7th Intern. Confer. on Very Large Data Bases*, Cannes, France, April 1981.

[GBLP96] J. Gray, A. Bosworth, A. Layman, and H. Pirahesh. Data cube: A relational aggregation operator generalizing group-by, cross-tab, and sub-totals. In *Proceed. 12th Intern. Conference on Data Engineering*, New Orleans, Louisiana USA, February 1996.

[Gho86] S.P. Ghosh. Statistical relational tables for statistical database management. *IEEE Transactions on Software Engineering*, SE-12(12):1106–1116, 1986.

[GMPS96] C. Glymour, D. Madigan, D. Pregibon, and P. Smyth. Statistical inference and data mining. *Communications of the ACM*, 39(11):35–41, 1996.

[GRT99] S. Grumbach, M. Rafanelli, and L. Tininini. Querying aggregate data. In *Proceedings of the Eighteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, May 31 - June 2, 1999, Philadelphia, Pennsylvania*, pages 174–184. ACM Press, 1999.

[Han97] J. Han. Olap mining: Integration of olap with data mining. In *Proceedings of the Eighth Working Conference on Database Semantics, Leysin, Switzerland*, 1997.

[Han98] J. Han. Towards on-line analytical mining in large databases. *SIGMOD Record*, 27(1):97–107, 1998.

[HRU96] V. Harinarayan, A. Rajaraman, and J.D. Ullman. Implementing data cube efficiently. In *Proceed. Intern. Conference on Management of Data - SIGMOD '96*, Montreal, Canada, 1996.

[LS97] H.-J. Lenz and A. Shoshani. Summarizability in olap and statistical data bases. In *Proceed. 9th Intern. Conference on Scientific and Statistical Database Management*, Olympia, Washington, USA, August 1997.

[McC82] L. McCarthy. Metadata management for large statistical databases. In *Proceed. of the 8th Intern. Confer. on Very Large Data Bases*, Mexico City, Mexico, September 1982.

[OOM85] G. Ozsoyoglu, Z.M. Ozsoyoglu, and F. Mata. A language and a physical organization technique for summary tables. In *Proceed. 4th ACM International Conference on Management of Data - SIGMOD '85*, Austin, Texas, May 1985.

[OOM87] G. Ozsoyoglu, Z.M. Ozsoyoglu, and V. Matos. Extending relational algebra and relational calculus with set-valued attributes and aggregate functions. *ACM Transactions on Database Systems*, 12(4):566–592, December 1987.

[Raf90] M. Rafanelli. *Encyclopedia of Computer Science and Technology, Vol. 23, Suppl. 8*, chapter Statistical and Scientific Database Management. A.Kent and J.G.Williams Ed.s, M.Dekker Pub., 1990.

[RBT96] M. Rafanelli, A. Bezenchek, and L. Tininini. The aggregate data problem: a system for their definition and management. *ACM Sigmod Record*, 25(4):8–13, December 1996.

[RR83] M. Rafanelli and F.L. Ricci. Proposal of a logical model for statistical data base. In *Proceed. 2th Intern. Workshop on Statistical Database Management*, Los Altos, California, September 1983.

[Sho82] A. Shoshani. Statistical databases: Characteristics, problems, and some solutions. In *Proc 8th Int Conf on Very Large Data Bases*, Mexico City, Mexico, September 1982.

[Sho97] A. Shoshani. Olap and statistical databases: Similarities and differences. In *Proceed. Intern. Confer. on Principles of Database Systems, PODS'97*, Tucson, Arizona, USA, May 1997.

[SR91] A. Shoshani and M. Rafanelli. A model for representing statistical objects. In *Proceed. of the 3rd Intern. Conference on Management of Data, COMAD '91*, Bombay, India, December 1991.

[Su83] S.Y.W. Su. Sam*: A semantic association model for corporate and scientific statistical databases. *Information Sciences*, 29(2-3):151–199, June 1983.

[SW85] A. Shoshani and H.K.T. Wong. Statistical and scientific database issues. *IEEE Transactions on Software Engineering*, SD-11(10):1040–1047, October 1985.