Querying Aggregate Data

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

Maurizio Rafanelli Leonardo Tininini
CNR-IASI
viale Manzoni 30
00185 Roma, Italy
{rafanelli,tininini}@iasi.rm.cnr.it

Abstract

We introduce a first-order language with real polynomial arithmetic and aggregation operators (count, iterated sum and multiply), which is well suited for the definition of aggregate queries involving complex statistical functions. It offers a good trade-off between expressive power and complexity, with a tractable data complexity. Interestingly, some fundamental properties of first-order with real arithmetic are preserved in the presence of aggregates. In particular, there is an effective quantifier elimination for formulae with aggregation.

We consider the problem of querying data that has already been aggregated in aggregate views, and focus on queries with an aggregation over a conjunctive query. Our main conceptual contribution is the introduction of a new equivalence relation among conjunctive queries, the isomorphism modulo a product. We prove that the equivalence of aggregate queries such as for instance averages reduces to it. Deciding if two queries are isomorphic modulo a product is shown to be NP-complete. We then show that the problem of complete rewriting of count queries using count views is also NP-complete. Finally, we introduce new rewriting techniques based on the isomorphism modulo a product to recover the values of counts by complex arithmetical computation from the views. We conclude by showing how these techniques can be used to perform automatic aggregation.

*Work done while the author was supported by IASI.

1 Introduction

The manipulation of aggregate data has gained considerable interest in recent years, for its great impact in various applications such as for instance data warehousing. In such applications, queries involve aggregation over evolving data of very large size. The use of materialized aggregate views, might strongly increase the efficiency of query processing.

The modeling and the manipulation of statistical data have been studied with different focus both in the field of statistical databases [Su83, SW85, OOM87, Gho86, RR93, RBT96], and in the field of on-line analytical processing (OLAP) [GBLP96, HRU96, LS97, Sho97]. 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. A fundamental problem of statistical databases, is to determine what can be derived from the statistical data. This problem, known as the *statistical inference* problem is fundamental both for restricting the derivable data for the protection of private data, as well as for deriving new data, by further aggregation of the statistical data, or by automatic aggregation.

In this paper, we present a first-order language for expressing general aggregate queries involving complex statistical functions. The language is based on real polynomial arithmetic together with aggregate operators that count, sum and multiply values in multisets. We first consider first-order logic with this signature $\mathrm{FO}_{\mathbb{R}}^{agg}$, and prove that every property that can be expressed with the aggregates can be expressed without aggregation. In other words, the logic $\mathrm{FO}_{\mathbb{R}}$ with polynomials over the reals, and its extension $\mathrm{FO}_{\mathbb{R}}^{agg}$ to aggregate functions coincide. This observation has fundamental consequences. In particular, there is an effective quantifier elimination method for formulae with real arithmetic and aggregation in $\mathrm{FO}_{\mathbb{R}}^{agg}$.

As a query language, it is easy to see, that with the relation symbols of the database schema, the aggregation operators provide additional expressive power.

For instance counting properties which are not definable with real arithmetic can be expressed. The aggregates allow the expression of rather complicated functions. In particular, they allow the definition of exponentiation with a small exponent. This can be used in practice to compute an average interest rate. Nevertheless, we prove that the query language $\mathrm{FO}_{\mathbb{R}}^{agg}$ enjoys a tractable data complexity.

We then consider the problem of querying aggregate views. In most cases it is undecidable if a query can be answered using aggregate views. This follows in particular from the undecidability of the type of constraints involved [RSSS98]. Sufficient conditions for the view usability problem have been shown in [SDJL96]. It is shown in particular that if only the query contains aggregation, there should be an isomorphism between the view and a part of the query. Some practical algorithms have been designed, among which the algorithm of [GHQ95] in a general setting, but which is not complete. In the present paper, we focus on decidable cases and complete techniques.

The present problem generalizes the classical problem of querying (non aggregate) views, which has been extensively studied in [LMSS95], where it was shown in particular that the complete rewriting of queries using views in the context where both queries and views are conjunctive with no comparison predicates is NP-complete.

An approach to the aggregate view usability problem consists in reducing it to a context with no aggregates. We consider queries which allow the application of one aggregate operation over a conjunctive query with neither comparison predicates nor constants, and see what reduces to the conjunctive query part. The equivalence of queries of this type has been studied previously in [NSS98], where it was shown that two count (resp. sum) queries are equivalent if their corresponding cores (the conjunctive query on which the aggregation is applied) are bag-set equivalent, thus reducing the problem of the equivalence of aggregate queries to a problem of equivalence of conjunctive queries. It is easy to see that the result also applies to multiply queries.

The main conceptual contribution of the paper is the definition of an equivalence relation between conjunctive queries, which appears to be fundamental in the presence of aggregate operators, called *isomorphism modulo a product*. Essentially, two queries are isomorphic modulo a product if they are set equivalent, and the multiplicity of the duplicates can be captured by additional literals.

We prove that the equivalence of aggregate queries such as average and percentage reduces to the isomorphism modulo a product of the core queries. This solves an open problem of [NSS98].

The complexity of checking the isomorphism modulo a product is shown to be NP-complete, thus similar to the classical equivalence problem. The equivalence of queries with arithmetic has been studied in [IS96], and shown to be of a much higher complexity than the present context.

We then see how these results can be used to solve the view usability problem. We consider the problem of the existence of a complete rewriting of a query using views. Given the correspondence between count and bag queries, we define a simple conjunctive rewriting of a bag query using bag views, similar to the set setting rewriting of [LMSS95]. We prove that the problem of complete rewriting for bag queries and views is NP-complete, a result similar to the set setting, but the techniques involved are rather different as shown in the paper.

The simple rewriting in the context of bags can be considered as to weak a method for solving the complete view usability problem. Indeed, there are cases where the query can be answered using the views with a complex arithmetic computation. We show another application of the isomorphism modulo a product, and introduce the rewriting modulo a product. This notion is shown to be more complex than the simple rewriting.

Finally, in the conclusion, we address the problem of the rewriting of the aggregate functions themselves, that is the rewriting of the aggregate function of the query in terms of the aggregate functions of the views in the context of a schema with a single relation. Elaborate techniques for reasoning with aggregation constraints have been presented in [LM96]. We prove that for a class of aggregates involving count, sum, average, and percent, the problem can be answered in polynomial time.

The paper is organized as follows. Section 2 presents the logic with aggregation and arithmetics over the reals. In Section 3, we consider the problem of equivalence of aggregate queries, and finally Section 4 is devoted to aggregate view usability.

2 Real aggregation

Standard aggregate (e.g. statistical) functions can be expressed by polynomial operations over the reals associated with aggregation over multisets of reals which *count, sum*, and *multiply* the elements. We study formally these operations in the context of first-order logic.

Multisets are necessary when considering aggregate functions. The language we consider though is restricted to sets, and the only use which is made of multisets is through the aggregation. For more on the expressive power of multisets, see [GLMW96]. Multisets are denoted by the symbol $\{\!\{\ \}\!\}$ when necessary.

2.1 First-order with aggregation

The mathematical context of the present setting 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 $\mathrm{FO}_{\mathbb{R}}$ first-order logic over this signature.

We introduce an extension of this signature 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 function over attributes $\bar{y}$ defined as follows: a tuple $\bar{a}$ is mapped to c_x iff c_x is the cardinality of $\{x|\varphi(x, \bar{a})\}$ if it is finite and $c_x = 0$ otherwise[1].
- **sum** $\Sigma_x^z\psi(x, \bar{y}, z)$ defines a function over attributes $\bar{y}$ defined as follows: a tuple $\bar{a}$ is mapped to s_x iff s_x is the sum of the elements of the multiset[2] $\{\!\{x|\psi(x, \bar{a}, z)\}\!\}$ if it is finite and non-empty, and $s_x = 0$ otherwise.
- **multiply** $\Pi_x^z\psi(x, \bar{y}, z)$ defines a function over attributes $\bar{y}$ defined as follows: a tuple $\bar{a}$ is mapped to p_x iff p_x is the product of the elements of the multiset[2] $\{\!\{x|\psi(x, \bar{a}, z)\}\!\}$ if it is finite and non-empty, and $p_x = 1$ otherwise.

The variable x is called the aggregated variable and z the multiset argument. The free variables $(\bar{y})$ of an aggregate term are the variables free in the formula which are neither aggregated nor used as multiset argument. 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. In the sequel we blur the distinction between a relation and a formula defining it, and between an attribute, and the variable defining it. So we allow the use of attribute names instead of variables as argument in the aggregation.

We consider the first-order logic $\mathrm{FO}_{\mathbb{R}}^{agg}$ over the signature of the real field with the three aggregation operators introduced above. The arithmetic operations are extended to functions, i.e. the sum and the product of two functions is a function. Properties on the number of roots of polynomials, their sum or product can be naturally expressed in this logic. Nevertheless, $\mathrm{FO}_{\mathbb{R}}^{agg}$ has no more expressive power than $\mathrm{FO}_{\mathbb{R}}$.

[1] Note that 0 denotes either the cardinality of an empty set, or of an infinite set. The semantics of 0 is therefore overloaded. In practice, it is easy to enforce the distinction explicitly.

[2] The multiplicity is related to the number of z values. More formally, s_x is the sum of the elements of the multiset $\{\!\{x \times C_z\psi(x, \bar{a}, z)|\exists z\psi(x, \bar{a}, z)\}\!\}$, and similarly for multiply.

Theorem 1 $\mathrm{FO}_{\mathbb{R}}^{agg} = \mathrm{FO}_{\mathbb{R}}$

The structure of the real field constitutes a good mathematical context for modeling aggregate data, in particular for its fundamental properties of o-minimality, decidable first-order theory, and quantifier elimination. A structure is *o-minimal* [DMM94] if every definable set, $\{x|\varphi(x)\}$, with φ a first-order formula, is a finite union of isolated points and open intervals. A structure admits *quantifier elimination* if for every first-order formula $\varphi(\bar{x})$, there exists an equivalent *quantifier free* formula $\psi(\bar{x})$.

Proof of Theorem 1 (sketch) The proof is made by induction on the imbrication of the second order operators and is based on the analytical properties of $\mathbb{R}$. Consider a formula

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

If $\bar{y} = \bar{a}$ is fixed, $\varphi(x, \bar{a})$ defines a finite union of isolated points and open intervals, by o-minimality. If there are open intervals,

$$C_x\varphi(x, \bar{a}) \equiv \bar{y} = \bar{a} \wedge n = 0$$

Otherwise,

$$C_x\varphi(x, \bar{a}) \equiv \bar{y} = \bar{a} \wedge n = \alpha$$

where α is the number of isolated points.

If $\bar{y}$ varies, there is a finite number of cells $\mathcal{C}_1, \cdots, \mathcal{C}_k$, defined by formulae $\varphi_1, \cdots, \varphi_k$, such that for each $i = 1..k$, the behavior of $\varphi(x, \bar{y}) \wedge \varphi_i(\bar{y})$ is stable with respect to the number of x's, and there is a constant α_i, equal to 0 if there are open intervals, and to the number of isolated points otherwise. The formulae $\varphi_1, \cdots, \varphi_k$ can be obtained from a cylindrical algebraic decomposition. Finally,

$$n = \overline{C_x\varphi(x,\bar{y})}(\bar{y}) \equiv \bigvee_i \varphi_i(\bar{y}) \wedge n = \alpha_i$$

The case of sum and multiply formulae is similar. In particular if $I_0 = \{i|\alpha_i = 0\}$ and $I_{>0} = \{i|\alpha_i > 0\}$ then

$$s = \overline{\Sigma_x\varphi(x,\bar{y})}(\bar{y}) \equiv \bigvee_{i \in I_0} \varphi_i(\bar{y}) \wedge s = 0 \vee$$

$$\bigvee_{i \in I_{>0}} \exists x_1 \cdots x_{\alpha_i}\ \varphi_i(\bar{y}) \wedge \varphi(x_1, \bar{y}) \wedge \cdots \wedge \varphi(x_{\alpha_i}, \bar{y})$$
$$\wedge x_1 < x_2 < \cdots < x_{\alpha_i} \wedge s = x_1 + x_2 + \cdots + x_{\alpha_i}$$

The induction is straightforward. $\square$

Theorem 1 has the fundamental corollary that these properties carry over in the presence of aggregate operators.

Corollary 2 There is an effective quantifier elimination method for formulae in $\mathrm{FO}_{\mathbb{R}}^{agg}$. Sentences can be decided, and monadic formulae define finite unions of isolated points and open intervals.

If we restrict $\mathrm{FO}_{\mathbb{R}}^{agg}$ formulae to involve only polynomials with integer coefficients, we can provide the following upper-bound on the complexity of the quantifier elimination.

Corollary 3 The quantifier elimination for $\mathrm{FO}_{\mathbb{R}}^{agg}$ can be performed in

$$AL(\log L)(\log\log L)(md)^{2^{O(w)}\ell\Pi_k n_k}$$

where A is the number of aggregate operations, L the bit length of the integer coefficients, m the number of polynomials, d their maximum degree, w the quantifier alternation, ℓ the number of free variables, and n_i the number of variables quantified by the ith vector of homogeneous quantifier ($\forall\cdots\forall$ or $\exists\cdots\exists$).

The proof follows directly from Renegar's complexity analysis [Ren92].

2.2 Aggregate query language

We now consider $\mathrm{FO}_{\mathbb{R}}^{agg}$ as a query language. We first recall the definition of databases. A *(database) schema* s is a finite set of relation symbols together with their arity, disjoint from the first-order language. We distinguish between predicates (such as $=, \leqslant$), and relations in s.

An *instance* I of s is a mapping from relation symbols to finite relations of the corresponding arity over $\mathbb{R}$.

Let $s = (R_1, \cdots, R_n)$ be a database schema. We denote by $\mathrm{FO}_{\mathbb{R}}^{agg}[s]$ first-order logic over the signature $\{\leqslant, +, \times, C, \Sigma, \Pi\} \cup s$. Any formula $\varphi(\bar{x})$ with free variables $\bar{x}$ in $\mathrm{FO}_{\mathbb{R}}^{agg}[s]$ defines a *query* mapping instances I of s to relations $\{\bar{a}|\mathbb{R} \cup I \models \varphi(\bar{a})\}$

It is immediate that $\mathrm{FO}_{\mathbb{R}}^{agg}$ has more expressive power than $\mathrm{FO}_{\mathbb{R}}$ as a query language, although $\mathrm{FO}_{\mathbb{R}}^{agg}$=$\mathrm{FO}_{\mathbb{R}}$ as a logic. In other words, for any database schema s,

$$\mathrm{FO}_{\mathbb{R}}[s] \ \subset \ \mathrm{FO}_{\mathbb{R}}^{agg}[s].$$

In particular, simple counting properties not expressible in $\mathrm{FO}_{\mathbb{R}}$, can be expressed such as cardinality comparisons. More generally, complex statistical properties can be expressed in $\mathrm{FO}_{\mathbb{R}}^{agg}$. The Min and the Max of a set of values, often given as primitives of query languages, can easily be defined, as well as complex statistical values such as for instance, the *median* value m of a finite set S of values which is defined by:

$$\overline{C_x(S(x)\wedge S(m)\wedge x\leqslant m)}(m) = \overline{C_x(S(x)\wedge S(m)\wedge x\geqslant m)}(m) + 0/1$$

$\mathrm{FO}_{\mathbb{R}}^{agg}$ also allows to express a limited form of exponentiation with the multiply operator. This can be used to express cumulated interest and average interest rates in financial applications for instance.

a

We next consider the relationship between the primitives of the language, and introduce some additional operators.

First it should be noted that the arithmetic operations $+$ and $\times$ can be expressed by the aggregate operators Σ and Π. Indeed, $x + y \equiv \Sigma_z(z = x \vee z = y)$, and $x \times y \equiv \Pi_z(z = x \vee z = y)$. Count can also be expressed by sum, $C_x\varphi(x, \bar{y}) = \Sigma_z^x(\varphi(x, \bar{y}) \wedge z = 1)$.

For practical purposes it is desirable to extend the set of aggregate operators. The new operators allow tuples in multiset and counting arguments, as well as (non-distinct) counters with multiset arguments with the obvious semantics. They are denoted as follows.

$$C_{\bar{x}}^{\bar{z}}\varphi(\bar{x}, \bar{y}, \bar{z}) \qquad \Sigma_x^{\bar{z}}\psi(x, \bar{y}, \bar{z}) \qquad \Pi_x^{\bar{z}}\psi(x, \bar{y}, \bar{z})$$

Proposition 4 The operations introduced above are definable in $\mathrm{FO}_{\mathbb{R}}^{agg}$.

The proof is based on equivalences of functional expressions which can be used for query rewriting in the context of views.

Let us now consider some examples of aggregate queries. We consider relations over an uninterpreted finite domain. The domain of the aggregate functions is assumed to be finite too.

Example 1 Consider a sample relation R over signature:

$$[\mathtt{Name}, \mathtt{Country}, \mathtt{Occupation}, \mathtt{Property}]$$

The attribute names are written with the letters N, C, O, P for short in the sequel. We consider the following queries.

Q_1 *percentage of people (with respect to the whole population of the country) by occupation and country*

It can be expressed by a formula $\varphi_1(a, c, o)$:

$$a = \frac{\overline{C_N^P R}(c, o)}{\overline{C_N^{OP} R}(c)}$$

$\overline{C_N^P R}$ is used as an abbreviation of $\overline{C_n^p R(n,c,o,p)}$. Note that the above fraction is used for readability. The expression should be written with $\times$. Nevertheless, it can be seen as a function mapping a tuple of values for c and o to a value a. The denominator is always different from 0.

Q_2 *average property of people by country and occupation*

It can be defined by $\varphi_2(a,c,o) : a = \dfrac{\overline{\Sigma_P^N R}(c,o)}{\overline{C_N^P R}(c,o)}$

Q_3 *(minimum) percentage of people whose global property is more than 50% of the entire property of the country*

We first define a subquery for the threshold value t such that the global property of people having a property strictly larger than t is less or equal to half of the global property, which itself is less or equal to the global property of people having a property larger or equal to t. The formula $\varphi(c,t)$ below defines the value of t for each country c:

$$\begin{aligned}
&\exists n, o\ R(n,c,o,t)\\
&\quad \wedge\ \overline{\Sigma_p^{n\,o}(R(n,c,o,p)\wedge p>t)}(c,t) \leqslant 0.5 \times \overline{\Sigma_P^{N\,O} R}(c)\\
&\quad \wedge\ \overline{\Sigma_p^{n\,o}(R(n,c,o,p)\wedge p\geqslant t)}(c,t) \geqslant 0.5 \times \overline{\Sigma_P^{N\,O} R}(c)
\end{aligned}$$

The query Q_3 can now be defined by $\varphi_3(c,a)$:

$$a = \frac{\overline{C_n^{opt}(R(n,c,o,p)\wedge\varphi(c,t)\wedge p>t)}(c)}{\overline{C_N^{OP} R}(c)}$$

The above queries can be equivalently written in an SQL dialect with "**select-from-where-group by-having**" syntax. For example,

```
Q1 select   country, occupation, a/b
   from     (select    country, occupation,
                                 COUNT(*) as a
             from      R
             group by  country, occupation)
            natural join
            (select    country, COUNT(*) as b
             from      R
             group by  country)
```

The data complexity of $FO_{\mathbb{R}}$ was investigated in the case of input consisting of finitely representable relations in [KKR90], where it was shown to be in NC. In the case of interest in the present study, with inputs restricted to finite relations, partial improvements of the NC bound have been obtained. It was shown in [BL96], that Boolean $FO_{\mathbb{R}}$ queries have TC^0 data complexity. In presence of aggregate operators, we are interested in non-Boolean $FO_{\mathbb{R}}$ queries, on which the aggregation is performed, and the previous result does not carry over. The result of aggregate queries over finite inputs might moreover be infinite, but is always finitely representable. This follows from the fact that there is quantifier elimination for $FO_{\mathbb{R}}^{agg}$ formulae (Corollary 2). The data complexity of $FO_{\mathbb{R}}^{agg}$ is also tractable as shown by the following result.

Theorem 5 $FO_{\mathbb{R}}^{agg}$ has NC data complexity.

Proof (sketch) The proof is made by induction on the imbrication of the aggregate operators. Each aggregate operator is replaced by a new variable, which represents the value of the function, and the aggregated formula is replaced by a formula without aggregation. The number of new variables introduced depends upon the query. Consider a formula

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

where φ involves database relations. The number of variables and quantifiers in the formula obtained by replacing each relation symbol in φ by its definition as a disjunction of equality atoms is independent of the instance. The formulae φ_i of the proof of Theorem 1, can be obtained in NC in the size of the instance, and therefore the rewriting of the previous formula to a formula without counting can also be performed in NC.

In the case of sum and multiply, one variable is introduced for the result of the aggregation, and $Sup(\alpha_i)$ existentially quantified variables to sum or multiply the values of x. The values α_i depend upon the polynomials appearing in the query.

So the number of variables and quantifiers, in the query or introduced to remove aggregation operators, depends only upon the query. $\square$

In the context of aggregate data, we are interested in finite relations. Queries in $FO_{\mathbb{R}}^{agg}$ are not guaranteed to produce finite outputs though. A query is said to be *safe* if it maps finite relations to finite relations. It is well known that the safety of relational calculus queries and hence $FO_{\mathbb{R}}^{agg}$ queries is undecidable.

However, it is easy to provide a syntactic characterization of safe queries, by checking that the cardinality of the output is finite. This can be done in $FO_{\mathbb{R}}^{agg}$. In fact the finitness is already expressible without aggregation by checking that the output has only isolated points, and so is finite (by o-minimality), a property expressible in $FO_{\mathbb{R}}$.

This does not result in an elegant language for safe queries though. In the following sections, we consider only safe formulae, and of a very restricted form, namely aggregation over conjunctive queries.

3 Query equivalence

In this section, we focus on the fundamental problem of query equivalence. Its undecidability in general, leads to restrict our study to simple classes of queries. We consider conjunctive queries without comparison predicates and constants. The restriction to a class of safe queries motivates the use of an active domain semantics for the interpretation of the aggregate functions. In the sequel, we therefore assume that aggregate functions, which always apply on finite inputs, are defined only on the domain of their input. Their graph is in turn a finite relation.

We consider aggregate relations with one summary attribute defined by functions of the following form:

$$\overline{Agg_y^{\bar{z}}(CQ(\bar{x},y,\bar{z}))}(\bar{x})$$

where Agg is some aggregation operation, and CQ is a conjunctive query over s with no constant and no comparison predicate.

An "*Agg-query*" is a query of this form with aggregation operator "Agg". We speak of *count queries* and *sum queries* for instance. The subquery CQ is called the core. This class has been studied in particular in [NSS98] where the equivalence of some Agg-queries, was reduced to relationships between the CQ queries.

Two equivalence relations are fundamental for conjunctive queries. Two conjunctive queries are *bag-set equivalent*, iff for each database instance, they produce the same result with the same multiplicity for each tuple [CV93]. Two queries are *(set) equivalent* iff for each database instance, they produce the same result independently of the multiplicity of the tuples. Another equivalence relation is fundamental in the context of bags. Two conjunctive queries are *bag equivalent*, iff for each *bag* database instance, they produce the same result with the same multiplicity for each tuple. It has been shown that two queries are bag-set equivalent iff they are isomorphic [CV93, NSS98]. It was shown in [NSS98] that two count (respectively sum) queries are equivalent iff their cores are isomorphic. The next proposition relates the different concepts of equivalence in the case of queries in canonical form [CV93]. A query is in *canonical form* if it does not contain duplicate literals.

Proposition 6 [CV93, NSS98] Given two conjunctive formulae $\varphi(\bar{x},\bar{y})$ and $\varphi'(\bar{x},\bar{y}')$ in canonical form, the following are equivalent:

1. the two count queries $\{[\bar{x},c] | c = \overline{C_{\bar{y}}\varphi(\bar{x},\bar{y})}(\bar{x})\}$ and $\{[\bar{x},c] | c = \overline{C_{\bar{y}'}\varphi'(\bar{x},\bar{y}')}(\bar{x})\}$ are equivalent;
2. the two queries $\{[\bar{x}] | \varphi(\bar{x},\bar{y})\}$ and $\{[\bar{x}] | \varphi'(\bar{x},\bar{y}')\}$ are isomorphic;
3. the two queries $\{\!\{[\bar{x}] | \varphi(\bar{x},\bar{y})\}\!\}$ and $\{\!\{[\bar{x}] | \varphi'(\bar{x},\bar{y}')\}\!\}$ are bag-equivalent.

Proposition 6 is used in Section 4 to rephrase the aggregate view usability problem in terms of conjunctive queries over bags. It thus constitutes a fundamental tool.

It is easy to see that a similar result holds for multiply queries.

Proposition 7 Two multiply queries are equivalent iff their cores are isomorphic.

The case of the average operator was left open in [NSS98]. For simplicity, let us write

$$Avg_y^{\bar{z}} = \frac{\Sigma_y^{\bar{z}}}{C_{y,\bar{z}}}$$

It is clear that if two conjunctive queries are isomorphic then their averages are equivalent, but the equivalence of average queries does not reduce to the isomorphism of the core queries. Indeed, consider the following example over a binary relation R.

$$q_1(x,y) :- R(x,y) \wedge R(x,z)$$
$$q_2(x,y) :- R(x,y)$$

The queries q_1 and q_2 are set equivalent, but not bag-set equivalent. Nevertheless, the corresponding average queries, $avg_1(x) = \overline{Avg_y^{z}(R(x,y) \wedge R(x,z))}(x)$ and $avg_2(x) = \overline{Avg_y(R(x,y))}(x)$ are equivalent. Indeed, for each x, if $c_i(x)$, and $s_i(x)$ denote the corresponding count and sum queries, we have $c_1(x) = (c_2(x))^2$, and $s_1(x) = s_2(x) \times c_2(x)$, and therefore

$$avg_1(x) = \frac{s_1(x)}{c_1(x)} = \frac{s_2(x)}{c_2(x)} = avg_2(x)$$

We next introduce a new equivalence relation between conjunctive queries, the *isomorphism modulo a product*.

Definition 1 Two conjunctive queries $Q_1(\bar{x},\bar{y}) :- b_1(\bar{x},\bar{y},\bar{z}_1)$ and $Q_2(\bar{x},\bar{y}) :- b_2(\bar{x},\bar{y},\bar{z}_2)$ over identical output variables, $\bar{x},\bar{y}$, are *isomorphic modulo a $\bar{y}$-uniform product* if they are set equivalent, and if there exist two conjunctions of literals $p_1(\bar{x},\bar{t}_1)$ and $p_2(\bar{x},\bar{t}_2)$, with the variables $\bar{t}_i$ distinct from the variables $\bar{z}_i,\bar{y}$ for $i \in \{1,2\}$, such that the following queries are isomorphic:

$$Q_1^{[q_1]}(\bar{x},\bar{y}) :- b_1(\bar{x},\bar{y},\bar{z}_1) \wedge p_1(\bar{x},\bar{t}_1)$$
$$Q_2^{[q_2]}(\bar{x},\bar{y}) :- b_2(\bar{x},\bar{y},\bar{z}_2) \wedge p_2(\bar{x},\bar{t}_2)$$

When it is clear from the context, we omit the uniformity with respect to the aggregated variables $\bar{y}$. The isomorphism modulo a product defines an equivalence relation between conjunctive queries. Note that two queries are isomorphic modulo a product only if they are set equivalent. Moreover, if the uniformity is with respect to an empty set of variables $\bar{y}$, the two notions coincide (it suffices to consider $p_1 = b_2$ and $p_2 = b_1$). The queries $q_1(x,y)$ and $q_2(x,y)$ of the above example are isomorphic modulo a product.

We can now state our main result for average queries.

Theorem 8 Two average queries are equivalent iff their cores are isomorphic modulo a product.

Note that the uniformity is with respect to the averaged variable.

The proof follows from the fact that two average queries are equivalent iff their corresponding count and sum queries have a similar behavior, which is precisely captured by the isomorphism modulo a product.

This relation seems fundamental when dealing with the equivalence of a large class of aggregate queries. In particular it applies to percentage queries. A *percentage query* is a query with aggregate operator percent (e.g. query Q_1 in Example 1). Let us write:

$$Perc_{\bar{y}}^{\bar{z}} = \frac{C_{\bar{z}}}{C_{\bar{y},\bar{z}}}$$

Theorem 9 Two percentage queries are equivalent iff their cores are isomorphic modulo a product.

In this case, the uniformity is with respect to the variable $\bar{y}$. The proof follows the same lines as the proof of Theorem 8.

Finally, we consider the complexity of the equivalence of the corresponding aggregate queries.

Theorem 10 The isomorphism modulo a product of conjunctive queries with neither comparison nor constant is NP-complete.

The NP-hardness is immediate. The membership in NP follows from the fact that the two queries p_1 and p_2 can be constructed non deterministically by choosing the appropriate literals in the body of the initial queries modulo an appropriate renaming of the variables.

It follows that the equivalence of respectively count, sum, multiply, average, and percent queries is decidable in NP. The case of count and sum was shown in [NSS98]. The case of multiply is similar. The cases of average, and percent follow directly from the previous result.

4 Querying aggregate views

We now consider the problem of aggregate view usability, for queries containing aggregates themselves. This problem is fundamental in the context of statistical databases, where the so-called macro data relations can be seen as aggregate views, as well as for OLAP systems, where the capability of answering a query using some precomputed (materialized) views, rather than accessing the entire relations, is crucial. Consider for instance the following query on the relation R of Example 1.

Q_4 *average property of people by country*

defined by $\varphi'_4(x,c) : x = \dfrac{\overline{\Sigma_P^{NO} R}(c)}{\overline{C_N^{PO} R}(c)}$

Assume that we are given two views R_1 and R_2 defined by the queries Q_1 and Q_2 of Example 1. The query Q_4 can be equivalently defined directly on the derived relations R_1 and R_2 by $\varphi''_4(x,c)$:

$$x = \overline{\Sigma_y^o(\exists p,a\ \ y = p \times a \wedge R_1(p,c,o) \wedge R_2(a,c,o))}(c)$$

The transformation of $\varphi'_4(x,c)$ into $\varphi''_4(x,c)$ is in general a difficult task that we consider in the sequel (see Theorem 16).

In general, information has been lost in the views, and it is undecidable if a query can be answered from the views. In some interesting cases the database contains enough information to answer a query, but the query cannot be expressed directly on the derived relations in the language, here $\mathrm{FO}_{\mathbb{R}}^{agg}$. This is the case in the following example.

Consider an initial schema with two monadic relations R and S respectively over attribute r and s, and two views, *product* $v_1 = \Pi_r R$ and *count* $v_2 = C_s S$. Then the query $q = \Pi_r^s(R \times S)$ whose value is $v_1^{v_2}$, is not definable in $\mathrm{FO}_{\mathbb{R}}^{agg}$ over the two views. This follows from Corollary 2, and the fact that the extension of the reals with exponentiation does not admit quantifier elimination [Dri82].

Before allowing the use of complex arithmetical operations to recover data from the views, consider a simple rewriting based on the correspondence between count and bag queries, which follows from Proposition 6. We define the problem of query rewriting in the bag context in a way analogous to that in the set context [LMSS95]. In the sequel all queries are conjunctive queries without constant and built-in predicates, and they apply on bag instances.

Definition 2 A bag query BQ' is a *rewriting* of a bag query BQ that uses the views $BV_1, \ldots, BV_n$ if:

- BQ and BQ' are bag-equivalent;
- BQ' contains literals among the BV_i's.

We are interested mostly in *complete rewriting*. A rewriting BQ' as above is complete if BQ' contains only literals among the BV_i's.

In the set context, the problem of finding a rewriting is closely related to containment mappings. The next example illustrates the novelty of the bag context.

Example 2 Consider the next query and views.

$q(x) : -p_1(x,y) \wedge p_2(y,z) \wedge p_3(z,w)$

$v_1(a,b) : -p_1(a,b) \wedge p_2(b,c)$

$v_2(d,e) : -p_2(d,e) \wedge p_3(e,f)$

By applying the algorithm for sets of [LMSS95], the query can be rewritten as

$q'(x) : -v_1(x,y) \wedge v_2(y,z)$

which is set but not bag equivalent to q. □

Unlike in the set context, the rewriting is required to be bag equivalent to the original query, or equivalently, the query obtained by replacing each view in the rewriting by its body is required to be bag-set equivalent to the original one. To achieve this goal, the literals of the query to be replaced by the view cannot be arbitrary with respect to the other literals of the query.

Consider a bag query $BQ : q(\bar{x}) : -\varphi(\bar{y})$ $(\bar{x} \subseteq \bar{y})$ and a bag view $BV : v(\bar{z}) : -\psi(\bar{z}, \bar{w})$, where $\bar{w}$ denotes the non-distinguished variables of the view and both φ and ψ are conjunction of literals. A homomorphism ϑ from the body of BV to the body of BQ, is a mapping of the variables of ψ to variables of φ such that each literal of ψ is mapped to a literal of φ. Let us denote by $\varphi'(\bar{y}')$ the conjunction of non mapped literals of φ, and call y' the *standing variables*.

Definition 3 A homomorphism ϑ from the body of a view BV to the body of a query BQ, is *sound* with respect to BQ and BV, if the non-distinguished variables of the view are disjoint from the standing variables of the query $(\vartheta(\bar{w}) \cap \bar{y}' = \emptyset)$.

states that a

We can now relate the rewriting in the bag context to the existence of sound homomorphisms.

Proposition 11 There is a rewriting of the bag query BQ using the bag view BV iff there exists a sound homomorphism ϑ from the body of BV to the body of BQ.

We have developed a sound and complete algorithm for finding the rewriting of a bag query using bag views. The algorithm replaces the literals in the query body by a view for which a sound homomorphism exists. The subtlety is that once a set of literals has been covered by a view, some of these literals cannot be covered again by other views, because this would violate the condition of soundness. The literals covered by a view containing only variables in the view head can still be used to ensure the match with another view body. Before giving the algorithm, we first illustrate the technique of replication of literals which can be used for another view in the next example.

Example 3 Consider the next query and views.

$q(x) : -p_1(x,y) \wedge p_2(y,z) \wedge p_3(y,w)$

$v_1(a,b) : -p_1(a,b) \wedge p_2(b,c)$

$v_2(d,e) : -p_1(d,e) \wedge p_3(e,f)$

We are looking for a rewriting that produces (after replacing each view literal by its body) an expression of the form:

$$(p_1(x,y))^i \wedge (p_2(y,z))^j \wedge (p_3(y,w))^h \text{ with } i,j,h \geqslant 1$$

where p^i represents i duplicates of the literal p.

A sound homomorphism exists from v_1 to q, so we can replace the corresponding literals in the body of q by v. The literal p_2 cannot be replicated because this would violate the soundness condition, while the literal p_1 can instead be replicated, so we can look for a rewriting that produces an expression of the form:

$$v_1(x,y) \wedge (p_1(x,y))^i \wedge (p_3(y,w))^h \text{ with } i \geqslant 0,\ h \geqslant 1$$

A sound homomorphism exists from v_2 to this partial rewriting of q, so we can rewrite the query as:

$$q'(x) : -v_1(x,y) \wedge v_2(x,y) \qquad \square$$

We now present the algorithm. The following notations will be used.

- QB a set containing all literals in the body of the query;
- AUX an auxiliary set (initially empty) containing all literals that are replicated during the rewriting process and may be used together with the remaining literals in QB to match one view body;
- SOL a set (initially empty) containing all views used in the rewriting;
- VB_i a set containing the literals in the body of the i-th view;
- $VB_i^{(h)}$ a subset of VB_i containing those literals, whose variables are all in the head of the view;

  i-th

  i-th view;
- $\vartheta(VB)$ the set VB, where all literal variables have been renamed according to the mapping ϑ;
- VH_i the head of the i-th view;
- N the number of views.

```
repeat
  choose i ⩽ N
  choose non deterministically a homomorphism
        ϑ from VB_i to QB ∪ AUX
  if ϑ is sound with respect to QB ∪ SOL
        and at least one literal l(ȳ) ∈ VB_i is
              such that l(ϑ(ȳ)) ∈ QB
  then
        QB = QB − (ϑ(VB_i) ∩ QB)
        AUX = AUX ∪ ϑ(VB_i^(h))
        SOL = SOL ∪ {VH_i(ϑ(ȳ))}
  endif
until QB = ∅
```

If at least one of the non-deterministic branches of the computation halts with $QB = \emptyset$, then the solution is in the corresponding SOL. Note that the loop need to be executed a number of times linearly bounded by the number of literals in the query body (at each step at least one literal of QB is eliminated). It is therefore easy to make it terminating.

The complexity of the complete rewriting problem in the bag setting is similar to the one in the set setting which was studied in [LMSS95].

Theorem 12 The problem of complete rewriting for bag queries and views is NP-complete.

Proof The principal source of complexity comes from the search of sound homomorphisms among a potentially exponential number. It is easy to check the soundness. On the other hand, a view is useful only if it covers at least one literal that has not been previously covered by any other view. So at each (non-deterministic) view replacement step the number of literals to be covered decreases. Hence, the number of view replacement steps is bounded by the number of literals in the query to be rewritten.

The complexity is determined by the non deterministic choice of the homomorphism ϑ, which is known to be NP (see [LMSS95] for the analogous problem in set rewriting). We can therefore conclude that the problem is in NP.

The NP-completeness follows for instance from the following observations on linear queries: (i) linear conjunctive queries are set-equivalent iff they are isomorphic [NSS98], and (ii) the complete rewriting problem for conjunctive (set) queries and views is NP-complete even when both the query and the views are linear [LMSS95]. So in this context finding a (bag) rewriting is equivalent to finding a (set) rewriting since the rewriting itself is linear. $\square$

It follows from Theorem 12, that the corresponding problem of complete rewriting for count queries and count views is also NP complete (note that the rewriting of count queries supposes a summation).

Corollary 13 The problem of complete rewriting for count queries and views is NP-complete.

The notion of rewriting introduced above for bag queries and views is a natural extension of the one in the set setting. It is not completely satisfying though in the context of bags. Indeed, when dealing with bags, we are interested in (i) the tuples which are in the output of the query like in the set setting, and (ii) their multiplicity. It is this last aspect that might require more care. The next examples show that in some cases the views provide enough information to answer the query, but arithmetical operations should be performed to recover the number of duplicates.

Example 4 Consider the next query and views.

$$q(x) :- p_1(x, y) \wedge p_2(x, z)$$
$$v_1(x) :- p_1(x, y) \wedge p_2(x, w_1) \wedge p_2(x, w_2)$$
$$v_2(x) :- p_2(x, v)$$

There is no complete rewriting of the query using the views given, however a complete "characterization" of q using only the views v_1 and v_2 can be provided.

First, note that the view v_1 is set equivalent to the query q. What needs to be characterized is the multiplicity of each tuple. For each tuple x, its multiplicity in respectively q, v_1, and v_2 is given by

$$c_q(x) = \overline{C_{yz}(p_1(x,y) \wedge p_2(x,z))}(x)$$
$$c_{v_1}(x) = \overline{C_{yw_1w_2}(p_1(x,y) \wedge p_2(x,w_1) \wedge p_2(x,w_2))}(x)$$
$$c_{v_2}(x) = \overline{C_v(p_2(x,v))}(x)$$

It is now immediate that $c_q(x) = \dfrac{c_{v_1}(x)}{c_{v_2}(x)}$.

So the multiplicity of each x can be obtained as the multiplicity of x in v_1 divided by the multiplicity of x in v_2. $\square$

The following example shows another arithmetical relation between the query and a view.

Example 5 Consider the following query and view.

$$q(x) :- p_1(x, y)$$
$$v(x) :- p_1(x, y) \wedge p_1(x, w)$$

It is clear that both are set equivalent. The multiplicity of each tuple in the query is the square root of its multiplicity in the view. $\square$

This leads to a more general notion of rewriting for bags which uses the isomorphism modulo a product.

Definition 4 Given a bag query $BQ : \{\!\{[\bar{x}] | \varphi(\bar{x}, \bar{y})\}\!\}$ and a collection of bag views $BV_1, \ldots, BV_n$, there is a *complete rewriting modulo a product* of BQ that uses the views $BV_1, \ldots, BV_n$ if there exists a number (*the power*) $\ell \in \mathbb{N}$, and two conjunctions of literals $p_1(\bar{z}_1)$ and $p_2(\bar{z}_2)$ such that:

1. the query $\{[\bar{x}] | p_2(\bar{z}_2)\}$ is set equivalent to BQ;
2. the queries $\{\!\{[\bar{x}] | \varphi(\bar{x}, \bar{y}_1) \wedge \ldots \wedge \varphi(\bar{x}, \bar{y}_\ell) \wedge p_1(\bar{z}_1)\}\!\}$ and $\{\!\{[\bar{x}] | p_2(\bar{z}_2)\}\!\}$ are bag-equivalent;
3. $\bar{y}_i \cap \bar{z}_1 = \emptyset$ and $\bar{y}_i \cap \bar{y}_j = \emptyset$ for $i \neq j \in \{1..\ell\}$;
4. both p_1 and p_2 contain only views in $BV_1, \ldots, BV_n$ as literals.

If $p_1(\bar{z}_1)$ and $p_2(\bar{z}_2)$ form a *complete rewriting modulo a product* of power ℓ of BQ that uses the views $BV_1, \ldots, BV_n$, then the tuples in the queries are those in p_2, and their multiplicity is given by

$$c_{BQ}(\bar{x}) = \sqrt[\ell]{\frac{c_{p_2}(\bar{x})}{c_{p_1}(\bar{x})}}$$

In Examples 4 and 5, there is a complete rewriting modulo a product of the query using the corresponding views. In Example 4, $p_1 = v_2$ and $p_2 = v_1$, with $\ell = 1$, and in Example 5, $p_1 = T$ and $p_2 = v$, with $\ell = 2$.

The complexity of the computation of a complete rewriting modulo a product is higher than the complexity of the computation of a simple rewriting as it follows from the next proposition.

Proposition 14 Given a bag query and a collection of bag views such that there is a complete rewriting modulo a product of the query that uses the views, the size of the two conjunctions of literals $p_1(\bar{z}_1)$ and $p_2(\bar{z}_2)$ forming the rewriting might be exponential in the size of the query and the views.

5 Conclusion and open problems

We have considered a query language to express aggregate queries which is based on the real field, and enjoys a tractable data complexity. Its expressive power which results from its group-by ability, seems to be well-suited for aggregation as can be shown through examples. Nevertheless, the definability in $\mathrm{FO}_{\mathbb{R}}^{agg}$ is not well understood yet. We conjecture that queries involving recursion (e.g. transitive closure of a finite graph) are not definable. Some results of this type are known for two-sorted languages [LW97]. It is also interesting to consider the expressive power resulting from the imbrication of aggregation. For a given schema s, does the nesting of aggregate operators defines a strict hierarchy of $\mathrm{FO}_{\mathbb{R}}^{agg}[s]$ queries?

We have then addressed the problem of query equivalence and the subsequent view usability problem. Our main conceptual contribution is to identify the notion of isomorphism modulo a product, which seems to play a fundamental role in this context. First, we show that we can reduce the equivalence of aggregate queries, such as averages, to the isomorphism modulo a product of the corresponding conjunctive queries, thus solving an open problem of [NSS98]. Second, we show that it is fundamental to recover the multiplicity of tuples for answering count queries using count views.

We have considered the case of aggregate queries with a core consisting of a conjunctive query with neither built-in predicates nor constants. Both additions result in general in a jump of complexity in the polynomial hierarchy for both the problem of query equivalence and the problem of complete rewriting. This is also the case in the bag setting.

The case of linear queries (that do not contain repeated predicates) is also interesting. Various problems become tractable. This is the case for the equivalence of conjunctive queries, and the result was extended to sum and count queries in [NSS98]. This carries over in the present setting, as shown by the next result which refines Theorem 10.

Proposition 15 The isomorphism modulo a product of linear conjunctive queries can be decided in polynomial time.

It follows that in this context the equivalence of average and percent queries can also be decided in polynomial time. On the other hand, the linearity does not result in a decrease of complexity for the problem of complete rewriting.

Finally, we have considered the case of a relational schema with a unique relation symbol R, where the problem of answering aggregate queries using aggregate views reduces to the ability of recomputing the aggregate function of the query using the aggregate functions of the views, thus performing automatic aggregation. We have proven that if the complete rewriting allows conjunction and nesting of aggregates, it can be solved in polynomial time.

Theorem 16 For a fixed schema with a single relation, and a query and views expressed by agg-queries for agg among $C, \Sigma, Avg, Perc$, the complete rewriting problem can be solved in polynomial time.

We have designed an algorithm which answers when possible the query using the views over a single relation. This algorithm derives automatically formula $\varphi_4''(x, c)$ from $\varphi_4'(x, c)$ of the begining of Section 4.

Acknowledgments

The authors wish to thank Leonid Libkin for helpful discussions on the reals with aggregation, and Werner Nutt for answering our questions on his paper.

References

[BL96] M. Benedikt and L. Libkin. On the structure of queries in constraint query languages. In *Proceedings 11th IEEE Symposium on Logic in Computer Science*, pages 25–34. IEEE Computer Society Press, 1996.

[CV93] S. Chaudhuri and M. Vardi. Optimization of real conjunctive queries. In *Proc. 12th ACM Symp. on Principles of Database Systems*, Washington, May 1993.

[DMM94] L. Van den Dries, A. Macintyre, and D. Marker. The elementary theory of restricted analytic fields with exponentiation. *Annals of Mathematics*, 85, 1994.

[Dri82] L. Van den Dries. Remarks on tarski's problem concerning $(R, +, \times, exp)$. In *Logic Colloquium*. Elsevier, North-Holland, 1982.

[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.

[GHQ95] A. Gupta, V. Harinarayan, and D. Quass. Aggregate-query processing in data warehousing environments. In *Proc. of Intl. Conf. on Very Large Data Bases*, 1995.

[GLMW96] S. Grumbach, L. Libkin, T. Milo, and L. Wong. Query languages for bags: Expressive power and complexity. *Sigact News*, 27(2):30–37, june 1996.

[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.

[IS96] O.H. Ibarra and J. Su. On the containment and equivalence of database queries with linear constraints. In *Proc. ACM Symp. on Principles of Database Systems*, 1996.

[KKR90] P. Kanellakis, G. Kuper, and P. Revesz. Constraint query languages. In *Proc. 9th ACM Symp. on Principles of Database Systems*, pages 299–313, Nashville, 1990.

[LM96] A.Y. Levy and I.S. Mumick. Reasoning with aggregation constraints. In *Proc. of Intl. Conf. on Extending Data Base Technology*, pages 514–534, 1996.

[LMSS95] A.Y. Levy, A.O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proc. ACM Symp. on Principles of Database Systems*, pages 95–104, 1995.

[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.

[LW97] L. Libkin and L. Wong. On the power of aggregation in relational query languages. In *Proc. 6th Int. Workshop on database programming languages*, 1997.

[NSS98] W. Nutt, Y. Sagiv, and S. Shurin. Deciding equivalence among aggregate queries. In *Proc. ACM Symp. on Principles of Database Systems*, pages 214–223, 1998.

[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.

[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.

[Ren92] J. Renegar. On the computational complexity and geometry of the first-order theory of the reals. *Journal of Symbolic Computation*, 13:255–352, 1992.

[RR93] M. Rafanelli and F.L. Ricci. Mefisto: a functional model for statistical entities. *IEEE Transactions on Knowledge and Data Engineering*, 5(4):670–681, August 1993.

[RSSS98] K.A. Ross, D. Srivastava, P.J. Stuckey, and S. Sudarshan. Foundations of aggregation constraints. *Theoretical Computer Science B*, 193(1-2):149–179, 1998.

[SDJL96] D. Sristava, S. Dar, H.V. Jagadish, and A.Y. Levy. Answering queries with aggregation using views. In *Proc. of Intl. Conf. on Very Large Data Bases*, pages 318–329, 1996.

[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.

[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.