On the content of materialized aggregate views

Stéphane Grumbach, Leonardo Tininini

On the Content of Materialized Aggregate Views

Stéphane Grumbach
INRIA
Rocquencourt BP 105
78153 Le Chesnay France
E-mail: Stephane.Grumbach@inria.fr

and

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

Version: revised Feb,20 2002

We consider the problem of rewriting queries using *only* materialized views. We first show that if the views subsume the query from the point of view of the information content, then the query can be rewritten using only the views, but the resulting query might be extremely inefficient. We then focus on aggregate views and queries over a single relation, which are fundamental in many applications such as data warehousing. We show that in this case, it is possible to guarantee that as soon as the views subsume the query, it can be rewritten in terms of the views in a simple query language. Our main contribution is the conception of rewriting algorithms which run in polynomial time, and the proof of their completeness which relies on combinatorial arguments. Finally, we consider the materialization of ratio views such as average and percentage, important for the design of materialized views.

1. INTRODUCTION

The manipulation of aggregate data is a fundamental issue in many applications such as data warehousing [Wid95, Gup97, MQM97], and OLAP systems [HRU96, AGS97, GM99b]. In such systems, queries involve aggregation over evolving data of very large size. The use of materialized aggregate views can strongly increase the efficiency of query processing.

Moreover, beyond the optimization problem, materialized views constitute sometimes the only data available, while the initial data cannot be accessed anymore. The practical importance of this problem is increasing for reasons related to data communication as well as to the type of the data, for instance with applications involving mobile computing [WSD+95, BI95], which often require to deal with the locally available data, as well as for instance statistical databases [OOM87, RBT96, Sho97], where the initial data often do not exist per se, or are made unavailable for reasons of privacy [MMR91, MM98].

In this paper we focus on the problem of rewriting queries into equivalent ones using *only* materialized views. This problem has been widely addressed in the literature for various classes of queries and views, such as in particular conjunctive

queries [LMSS95, CKPS95], and aggregate queries [SDJL96, CNS99, GRT99]. Algorithms have been designed to rewrite the initial queries against the database into equivalent queries against the views.

An alternative way to process queries using views makes an explicit use of the view extensions in addition to their definitions. View-based query answering allows to retrieve certain answers to the query that comply with any database instance coherent with the view extensions.

Query answering can in general be more effective than query rewriting. Indeed, it might allow the user to derive answers even when an equivalent rewriting does not exist. Equivalent query rewriting though is promising in the present context. First, the complexity of computing the equivalent rewriting depends only upon the size of view expressions, and not upon the size of the data, which is supposed to be large in the applications considered. More importantly, when aggregating, the answer to a query should be exact. Incomplete answers to queries would lead to approximate aggregation, a topic of strong interest, but beyond the scope of this paper. Finally, we focus in this paper on the completeness with respect to the information content of the views of the rewriting methods we propose, thus restricting to equivalent rewriting.

To the best of our knowledge, the problem of deciding if views contain enough information to rewrite a query has not been addressed. Deciding if the views allow the rewriting of a query is a fundamental question both positively for answering the query if no other data are available, and negatively, for protecting data with the guarantee that the views do not permit to derive sensitive information.

We introduce a notion of completeness which is not restricted to any syntactic type of rewriting. A rewriting algorithm will be said to be complete if intuitively arbitrarily more complex forms of rewriting would not lead to any improvements.

We characterize the information content of collections of views or queries, by their capability to distinguish different database instances. We say that a collection of queries, $\mathcal{Q}$, subsumes another one, $\mathcal{Q}'$, if $\mathcal{Q}$ has at least the distinguishing power of $\mathcal{Q}'$, or, more formally, if for each pair of instances I_1, I_2, if they are undistinguishable by $\mathcal{Q}$ (same answers for all queries on the two instances), they are undistinguishable by $\mathcal{Q}'$.

Our first result shows that a query can be rewritten using a collection of views iff the views subsume the query. The resulting rewritten query might be extremely inefficient. We then consider rewritings restricted to fixed query languages, as is usually done. It is easy to see though that in general a restricted rewriting results in incomplete methods, that fail to rewrite in presence of enough information. In the literature, weaker forms of completeness are often proposed relative to some specific query languages in which the rewritten queries are expressed.

We then turn to aggregate queries and views, and focus on a class defined by one aggregation over one relation. Although this restriction might seem drastic, these queries are fundamental in many applications in OLAP and data warehousing [GBLP96, HRU96, GHRU97], where data are first extracted according to specified criteria and then aggregated along several "dimensions" to analyze. *Iceberg* queries and cubes are also based on a similar assumption [FSGM+98, BR99].

We consider aggregate functions such as *count*, and iterated *sum* and *multiply*, and ratio of them: average *avg*, and percentage *perc*. We propose a rewriting algorithm **AggRew**, for rewriting queries in terms of *count*, *sum* and *multiply* views. The rewriting techniques we propose for aggregate queries and views are similar (in the COUNT and SUM part) to those proposed in [SDJL96, CNS99, GRT99], but

extend them to the processing of percentage and average queries/views.

The main result of the paper is to show that the proposed algorithm is sound and complete with respect to the information content, thus showing that no better rewriting can be obtained, no matter how complex and rich the rewriting technique and the target query language are. So, the algorithm **AggRew** produces a rewriting iff the views subsume the query. We believe that this is the first result of this type. The proof of completeness is the subtle part. It relies on combinatorial lemmas showing that the possible rewritings are in fact restricted. The complexity of the algorithm is shown to be linear.

We then consider the case of views with ratio aggregations such as *average* and *percentage* (stored as ratio). In this case, we have shown that, even if some manipulations may produce an exponential burst of the computation, a polynomial time algorithm, **AggRew2**, can be obtained. The strategy used to reduce the complexity relies on a separation of rewriting rules that are conservative for the information content, and rules that rewrite views into views with less information content. The former are shown to produce only a quadratic blow up, while the latter might produce an exponential blow up. We show that the latter can be used at the end of the computation, and that one application of these rules is sufficient.

We have shown that the algorithm **AggRew2** is sound, but we were unable to prove its completeness with respect to the information content. We have demonstrated a weaker form of completeness with respect to a set of elementary rewriting rules. The main interest of this second algorithm is the tractability which can be maintained although the search space has become exponential with the ratios. Moreover, it is shown that by an increase in the number of materialized views, a linear algorithm can be obtained.

These results provide insight in database schema design. Indeed, the choice of the views to be materialized determines how efficient the rewriting process is and which queries can be rewritten. This can be used to make data accessible, or on the contrary to protect sensitive data. Note that we consider only the problem of rewriting queries using views, and that the protection of data has to be addressed mainly at the level of instances.

The paper is organized as follows. In the next section, we present related work on query processing using views. We then present two motivating examples with aggregate views, and analyze the possibility of rewriting queries using the views. In Section 4, we introduce the concept of subsumption between collections of queries. In Section 5, aggregate queries are introduced, and the notations are illustrated. The algorithm for rewriting queries from flat views is presented in Section 6. It is shown to be complete and to run in linear time. Finally, in Section 7, we consider ratio materialized views.

2. RELATED WORK

The problem of computing the answer of a query by using some precomputed (materialized) views has been extensively studied in the literature and has been generically denoted as the problem of *answering queries using views* (see for instance [LMSS95, DG97, Hal00]).

Informally speaking the problem can be stated as follows: given a query Q and a collection of views $\mathcal{V}$ over the same schema s, is it possible to compute the answer of Q by using (only) the information provided by the views in $\mathcal{V}$?

A more rigorous distinction has been made between *view-based query rewriting* and *query answering*, corresponding to two distinct approaches of the general problem [CGLV00d, CGLV00c, Hal00]. Query rewriting is based on the use of the view definitions to produce a new rewritten query, expressed in terms of the available view names. The answer of the query can then be obtained by using the rewritten query and the view extensions. Query answering is based instead on the exploitation of both the view definitions and the view extensions, trying to determine the best possible answer, possibly a subset of the exact one, which can be extracted from the view extensions. More precisely, the goal is to compute the *certain answers* to the query with respect to the views, i.e. the set of tuples that are in the answer of the query for every database instance, which is compatible with the view extensions.

Several authors have studied the query answering problem for various classes of queries and views, both under the Closed and the Open World Assumption [AD98, GM99a, CGLV00a, CGLV00b]. Generally speaking, query answering techniques are preferable in contexts where exact answers are rather unlikely to be obtained (e.g. integration of heterogeneous data sources, like Web sites), and when requirements on response times are not very stringent. On the other hand, as noted in [GM99a], query answering methods can be extremely inefficient, because it is difficult or even impossible to process only the "useful" views and to apply optimization techniques such as pushing selections and joins. As a consequence, the rewriting approach is more appropriate in contexts like OLAP systems, where the amount of data is very large and fast response times are required [GL01], and for query optimization, where different query plans need to be maintained in main memory and efficiently compared [ALU01].

Moreover, an important distinction has to be made between algorithms for equivalent (or exact) rewritings and algorithms to produce maximally contained (or perfect) rewritings. The former produce a rewritten query, which is equivalent to the original one, while in the latter case the rewritten query is the best contained approximation of the original one, which can be obtained from the available views. The complexity of answering/rewriting queries using materialized views depends upon the syntax of the views and the queries [DG97, AD98, AGK99].

Several algorithms have been proposed to produce equivalent rewritings when the query and the views are conjunctive (but do not contain aggregations). Particularly, the bucket algorithm [LRO96], the inverse-rule algorithm [Qia96], the Shared-Variable-Bucket algorithm [Mit99] and MiniCon [PL00] (which are both refinements of the bucket), and CoreCover [ALU01]. The idea of the inverse-rule has also been used to produce maximally contained rewritings [DG97, AGK99]. In [LRU96] the authors propose an algorithm for equivalent rewriting in case of an infinite number of views.

Some algorithms have also been proposed to produce equivalent rewritings, when the query and the views are conjunctive and they both contain the standard aggregations COUNT, SUM, MIN and MAX [SDJL96, CNS99, GRT99]. Particularly, the algorithm proposed by Cohen et al. is based on the following technique: (i) each of an exponential number of "candidate rewritings" is considered; (ii) the candidate rewriting is unfolded (i.e. the view literals are replaced by their body expressions); (iii) finally, the equivalence of the obtained query and the query to rewrite is checked by using known equivalence criteria. The three steps are repeated until a successful (equivalent) rewriting is detected or the set of candidate rewriting has been exhausted, thus indicating a failure.

The first algorithm we propose in this paper has some analogies (in the COUNT and SUM part) with the one proposed in [CNS99], since it is also based on rewriting count queries using count views and on rewriting sum queries using either sum or count views. However, the focus of our paper is very different, since we only consider polynomial algorithms (although obviously assuming a more restricted form for queries and views), we also consider the impact of ratio aggregations, namely averages and percentages, and above all we prove that the algorithm for "flat" views is complete with respect to the information content, i.e. that no better rewriting can be obtained, no matter how complex and rich the rewriting technique and the target query language are.

In fact, all algorithms proposed in the literature produce rewritten queries of a particular syntactic form (typically a conjunction of view literals or a disjunction of conjuncts), but there is no guarantee that if the algorithm fails, an equivalent rewriting could not be obtained by considering a more general (or simply different) strategy and/or language to rewrite.

Let us consider for instance the algorithm presented in [DG97, DGL00], which generates "maximally contained" rewritings. Given a query Q and a collection of views $\mathcal{V}$, the algorithm produces a rewritten query Q' and it is shown that *any* other rewriting Q'' which is contained in Q has also to be contained in Q', thus showing maximality. However, the containment of the arbitrary Q'' in Q' is proven by using the containment algorithm proposed in [RSUV89], thus implicitly assuming that Q'' is non-recursive. Containment of recursive queries is known to be undecidable [Shm87]. Thus, the previous proof technique can only lead to maximality in some restricted set of queries expressible in some fixed query language where containment is decidable.

The use of materialized views with aggregations has been shown to be fundamental in the context of OLAP systems and data warehousing [GBLP96, AGS97, CT97, CD97]. In such applications, the efficient use of precomputed views is often the only way to obtain acceptable (On Line) response times. Several authors have proposed techniques to select a "good" set of aggregate views to materialize [HRU96, Gup97, GHRU97, BPT97] and when the query cannot be easily rewritten in terms of the materialized views, queries are usually computed directly on the fact table by means of specialized indexing structures [OQ97, CI98, JL01].

3. MOTIVATIONS

We consider the problem of rewriting queries by using views expressed as simple aggregations over the same relational expression. The following examples, expressed in SQL-syntax, illustrate the technical problem.

EXAMPLE 1. The data warehouse of a company contains some precomputed views extracted from its sale database. Among them, we consider the main relation R in the database of schema:

```
(Product,Color,State,Date,Qty)
```

which contains an already aggregated version of the sale database. Consider the following query.

select Product, Color, **sum**(Qty)
from R
group by Product, Color

This query can be easily answered by using only the following view, without accessing the original relation,

create view V1 **as**
select Product, Color, Date, **sum**(Qty) **as** Q
from R
group by Product, Color, Date

the rewritten (equivalent) query being:

select Product, Color, **sum**(Q)
from V1
group by Product, Color

It is less obvious however whether the query can be rewritten using the following views:

create view V2 **as**
select Product, State, Date, **sum**(Qty) **as** Q
from R
group by Product, State, Date

create view V3 **as**
select Color, State, Date, **sum**(Qty) **as** Q
from R
group by Color, State, Date

The fact that the two views have a high level of detail and the presence in the views of the attributes required by the query may suggest that such extraction is possible. However, we will prove that it is not the case. □

In the previous example, Q has been obtained by "removing" the attribute Date from $V1$. Such an operation is usually called *summarization*, and also, in the OLAP terminology, *roll-up* [Sho97]. In the following we will generally refer to it by the former term and say for instance that Q can be obtained by summarizing the attribute Date of $V1$. We next consider an example of percentage query rewriting.

EXAMPLE 2. A mobile phone company maintains a data warehouse containing the details on the traffic among its customers and particularly a table (relation) `InterCalls` on the schema

`(Call-id,Cust,Day,Source,Dest,Plan,Dur)`

where Call-id is the identifier of the call, Cust is the calling customer, Day is the day of the call, Source is the antenna to which the customer is connected when the call starts, Dest, the antenna to which the called customer is connected when the call starts, Plan is the calling plan of the customer, and Dur is the duration of the call.

Some aggregations are applied on this table, to compare the use of the antennas, guarantee an optimal workload, and propose new calling plans, according to the previous aspects. In particular, the following view has been precomputed:

```
create view V1 as
select  Day, Source, Dest, Plan, N/D
from    ((select    Day, Source, Dest, Plan, count(*) as N
          from      Intercalls
          group by  Day, Source, Dest, Plan)
          natural join
          (select   Day, Source, count(*) as D
          from      Intercalls
          group by  Day, Source))
```

V1 represents the percentage of phone calls by day, calling plan, source and destination antennas, with respect to the total number of calls for the same day and source antenna. We suppose that further studies are required and that the following queries are posed to the DW system.

The problem we address is whether the queries can be rewritten using only the view $V1$ and without accessing the table `InterCalls`.

Query Q1:

```
select  Day, Source, Plan, N/D
from    ((select    Day, Source, Plan, count(*) as N
          from      Intercalls
          group by  Day, Source, Plan)
          natural join
          (select   Day, Source, count(*) as D
          from      Intercalls
          group by  Day, Source))
```

Query Q2:

```
select  Day, Source, Dest, Plan, N/D
from    ((select    Day, Source, Dest, Plan, count(*) as N
          from      Intercalls
          group by  Day, Source, Dest, Plan)
          natural join
          (select   Day, count(*) as D
          from      Intercalls
          group by  Day))
```

Query Q3:

```
select  Day, Source, Dest, Plan, N/D
from    ((select    Day, Source, Dest, Plan, count(*) as N
          from      Intercalls
          group by  Day, Source, Dest, Plan)
          natural join
          (select   Day, Source, Plan, count(*) as D
          from      Intercalls
          group by  Day, Source, Plan))
```

We show in particular that the queries $Q1$ and $Q3$ can be computed by using the view $V1$ and propose an algorithm to rewrite them, while we show that the information content of $V1$ is not sufficient by itself to rewrite the query $Q2$. Finally, let us consider a new view $V2$:

```
create view V2 as
select  Day, Source, Dur, N/D
from    ((select    Day, Source, Dur, count(*) as N
          from      Intercalls
          group by  Day, Source, Dur)
          natural join
          (select   Day, count(*) as D
          from      Intercalls
          group by  Day))
```

V2 seems to have no influence on the answerability of the above queries. However, we show that $V1$ and $V2$ *together* allow us to compute the query $Q2$. □

4. THE INFORMATION CONTENT

In this section, we formally consider the following question: What information is contained in a view? and the related problem: Given a collection of views, can a query be equivalently rewritten using the views? We address the question from the point of view of the information content.

For that purpose, we introduce a concept of *subsumption* between collections of queries. In other words, we characterize the information content of a collection of queries by its distinguishing power, i.e. by its capability to determine that two database instances are different.

We first briefly recall the definitions of databases and queries. A *(database) schema* s is a finite set of relation symbols together with their arities. We consider as universe the set of rational numbers $\mathbb{Q}$ (in practice, an additional non numerical universe can be assumed too). An *instance* I over s is an interpretation of the relation symbols of s by finite relations of the corresponding arity over $\mathbb{Q}$. A *query* (or *view*) over s is a partial recursive mapping from instances over s to relations of some fixed arity. We do not make any assumption of genericity on the queries [AHV95], since we consider for simplicity only a numerical domain. Clearly the following results can be extended to a framework with a non-numerical domain and queries generic with respect to the latter universe.

A view is a query to which a name has been given. Its name will generally be the predicate under which the resulting relation is stored. We will also use the predicate symbol to denote the view expression, or the mapping defined by the view. The meaning will be clear from the context.

Let $\mathcal{Q} = (Q_1, \cdots, Q_\ell)$ be a collection of queries over some schema s, and let I be an instance over s. We denote by $\mathcal{Q}(I)$ the collection of relations $(Q_1(I), \cdots, Q_\ell(I))$. We can now define the *subsumption* between two collections of queries.

DEFINITION 1. Let $\mathcal{Q}$ and $\mathcal{Q}'$ be two collections of queries over some schema s. We say that $\mathcal{Q}$ *subsumes* $\mathcal{Q}'$, denoted $\mathcal{Q} \models \mathcal{Q}'$, iff for each pair, I_1, I_2, of instances over s, $\mathcal{Q}(I_1) = \mathcal{Q}(I_2)$ implies $\mathcal{Q}'(I_1) = \mathcal{Q}'(I_2)$.

The relation of subsumption defines a semilattice over collections of queries. Let $\mathbf{Q}_s$ be the set of all queries over schema s, and $\mathcal{P}(\mathbf{Q}_s)$, the set of all subsets of $\mathbf{Q}_s$. The subsumption relation defines a preorder over $\mathcal{P}(\mathbf{Q}_s)$. We introduce the corresponding equivalence relation $\approx$ defined by $\models \wedge \dashv$.

We consider the structure $(\mathcal{P}(\mathbf{Q}_s)/\approx, =|, \cup)$, with the relation *subsumed-by*, $=|$, defined as follows. Consider two elements S, and S' of $\mathcal{P}(\mathbf{Q}_s)$, and let us denote by $\bar{S}$, and $\bar{S}'$ the corresponding elements in $\mathcal{P}(\mathbf{Q}_s)/\approx$. Then $\bar{S} =| \bar{S}'$ iff $S =| S'$, and $\bar{S} \cup \bar{S}' = \overline{S \cup S'}$. It is easy to see that the union operation defines the *least upper bound* of two sets of queries for the subsumed-by relation. The greatest element is $\mathbf{Q}_s$, while $\emptyset$ is the smallest one.

PROPOSITION 1. *$(\mathcal{P}(\mathbf{Q}_s)/\approx, =|, \cup)$ is a join semilattice.*

Proof. Consider two elements S, and S' of $\mathcal{P}(\mathbf{Q}_s)$. Clearly, $S \cup S' \models S$, and $S \cup S' \models S'$. Let S'' be such that $S'' \models S$, and $S'' \models S'$. We want to prove that $S'' \models S \cup S'$. That is for all instances I_1, I_2 over s, $S''(I_1) = S''(I_2) \Rightarrow (S \cup S')(I_1) = (S \cup S')(I_2)$.

Assume that I_1, I_2 are such that $S''(I_1) = S''(I_2)$. It follows that $S(I_1) = S(I_2)$, and $S'(I_1) = S'(I_2)$. Therefore for each query q in $\bar{S}$, and each query q in $\bar{S}'$, $q(I_1) = q(I_2)$. Since $\overline{S \cup S'} = \bar{S} \cup \bar{S}'$, it follows that for all queries q in $\overline{S \cup S'}$, $q(I_1) = q(I_2)$, and therefore, $(S \cup S')(I_1) = (S \cup S')(I_2)$. ∎

Proposition 1 plays a fundamental role in the proofs of completeness of the rewriting algorithms, as it greatly reduces the number of views to be considered. It is used in particular in the proof of the lemmas for Theorem 4.

We now consider the related view problem. Given a collection of views $\mathcal{V}$ over some schema s, and an instance I over s, we denote by $I^{\mathcal{V}}$ the extension of I to the views in $\mathcal{V}$, with the views interpreted by their results on I. Two queries q and q' over relation symbols in $s \cup \mathcal{V}$ are *equivalent modulo* $\mathcal{V}$ [CNS99] if for any instance I over s, $q(I^{\mathcal{V}}) = q'(I^{\mathcal{V}})$.

DEFINITION 2. Given a collection of views $\mathcal{V}$ and a query q over schema s, a query q' over $\mathcal{V}$ is an *equivalent rewriting of q using $\mathcal{V}$* if q and q' are equivalent modulo $\mathcal{V}$.

Note that the rewriting is not related to any target query language. In the following we only consider equivalent rewritings and we refer to them by the term rewritings for brevity. We say that a query q can be rewritten using $\mathcal{V}$, if there is an *equivalent rewriting of q using $\mathcal{V}$*.

The next theorem relates the problem of rewriting using views to the information content of the views.

THEOREM 2. *Let $\mathcal{Q}$ and $\mathcal{Q}'$ be two collections of queries over some schema s. Then the following are equivalent:*

(i) for all $q \in \mathcal{Q}'$, q can be rewritten using $\mathcal{Q}$

(ii) $\mathcal{Q} \models \mathcal{Q}'$

Proof. $(i) \Rightarrow (ii)$: there is a query q' over $\mathcal{Q}$ which is equivalent to q modulo $\mathcal{Q}$. Consider two instances I_1, I_2 over s. If $\mathcal{Q}(I_1) = \mathcal{Q}(I_2)$, then $q'(I_1^{\mathcal{Q}}) = q'(I_2^{\mathcal{Q}})$, and hence $q(I_1) = q(I_2)$. If this holds for all $q \in \mathcal{Q}'$, it follows that $\mathcal{Q} \models \mathcal{Q}'$.

$(ii) \Rightarrow (i)$: Conversely, if $\mathcal{Q} \models \mathcal{Q}'$, let $q \in \mathcal{Q}'$. We define a query q' over $\mathcal{Q}$ which for each instance I, maps $\mathcal{Q}(I)$ to $q(I)$ as follows. Given an enumeration of the instances over s, let J_{min} be the first instance such that $\mathcal{Q}(I) = \mathcal{Q}(J_{min})$. The result of the query is $q(J_{min})$. ∎

The previous theorem shows that if there is enough information in the views, there is a rewriting. Consider two collections of queries $\mathcal{Q}$, and $\mathcal{Q}'$. The fact that $\mathcal{Q}$ does not subsume $\mathcal{Q}'$, implies that there are instances I_1, I_2 which can be distinguished by $\mathcal{Q}'$, but are undistinguishable by $\mathcal{Q}$. It follows that (at least some) queries in $\mathcal{Q}'$ cannot be rewritten using $\mathcal{Q}$, since $\mathcal{Q}'(I_1)$ and $\mathcal{Q}'(I_2)$ are distinct, while $\mathcal{Q}(I_1)$ and $\mathcal{Q}(I_2)$ are identical. On the other hand, if $\mathcal{Q}$ subsumes $\mathcal{Q}'$, $\mathcal{Q}$ has a distinguishing power at least equal to that of $\mathcal{Q}'$, and it suffices to rewrite the queries in Q'.

A *rewriting algorithm* is an algorithm which, given as input a collection of views $\mathcal{V}$ and a query q over schema s, produces as output a rewriting q' of q using $\mathcal{V}$, if it exists, or a failure message, otherwise.

For a given algorithm α, we will write $\mathcal{V} \vdash_\alpha q$ whenever the algorithm produces a rewriting of q using $\mathcal{V}$. A rewriting algorithm is *sound* if for each collection of views $\mathcal{V}$ and each query q, $\mathcal{V} \vdash_\alpha q \Rightarrow \mathcal{V} \models q$. It is *complete* if for each collection of views $\mathcal{V}$ and each query q, $\mathcal{V} \models q \Rightarrow \mathcal{V} \vdash_\alpha q$. (Note that this notion of completeness should be distinguished from the fact that the rewriting produces a query involving only the views and no relations of the initial schema as in [LMSS95].) Similarly, we will consider rewriting systems, and speak of their soundness and completeness, meaning that there is a derivation producing a rewriting q' of q using $\mathcal{V}$.

Theorem 2 is only of theoretical interest, and in practice, for reasons of efficiency, rewriting algorithms produce queries of a restricted form. The objective is that (i) the algorithm is complete, (ii) the rewritten query can be obtained efficiently, and (iii) it can be evaluated efficiently as well. We consider these aspects in the sequel, by studying the completeness and complexity of rewriting algorithms, restricted to queries and views in fixed query languages.

Note that in general if the query language used to express the rewritten query is restricted, this may lead to incomplete rewriting algorithms. For instance it is easily shown that, if queries and views are conjunctive with comparison predicates, then a rewriting algorithm that considers only rewritten queries of the same form is incomplete: in order to rewrite the query $q() \leftarrow r(x)$ using the views $v_1(x) \leftarrow r(x) \wedge x > 0$, $v_2(x) \leftarrow r(x) \wedge x \leqslant 0$ at least a query language with disjunction is needed, the rewritten query being $q'() \leftarrow v_1(x) \vee v_2(x)$. In the next sections, we present classes of queries and views such that a complete rewriting algorithm exists, even though using a restricted query language to rewrite.

5. AGGREGATE QUERIES

We now turn to a restricted class of queries, that include the queries of Examples 1 and 2, and can be expressed with the following aggregate operators [GRT99], defined over multisets which *count, sum*, and *multiply* the elements. The operators sum and multiply are defined over the rationals. Let ψ be a relational expression (e.g. a relation). We assume that variables correspond to specific attributes of the relational expression independently of their written position. The correspondence is made explicit only when necessary.

- **count** $C^{\bar{z}}\psi(\bar{x}, \bar{z})$ is a partial function mapping a tuple $\bar{a}$ such that $\exists \bar{z}\psi(\bar{a}, \bar{z})$, to the cardinality c_a of the set $\{\bar{z}|\psi(\bar{a}, \bar{z})\}$.

- **sum** $\Sigma_y^{\bar{z}}\psi(y,\bar{x},\bar{z})$ is a partial function mapping a tuple $\bar{a}$ such that $\exists y\bar{z}\psi(y,\bar{a},\bar{z})$, to the sum s_a of the elements of the multiset[1] $\{\!\{y|\psi(y,\bar{a},\bar{z})\}\!\}$.

- **multiply** $\Pi_y^{\bar{z}}\psi(y,\bar{x},\bar{z})$ is a partial function mapping a tuple $\bar{a}$ such that $\exists y\bar{z}\psi(y,\bar{a},\bar{z})$, to the product p_a of the elements of the multiset[1] $\{\!\{y|\psi(y,\bar{a},\bar{z})\}\!\}$.

For example, based on the relation $R(i,c,x,s,y,p,z)$ introduced in Example 2, with schema `(Call-id,Cust,Day,Source,Dest,Plan,Dur)`,

count $C^{icsypz}R(icxsypz)$ defines a function mapping each day x to the total number of calls made in that day;

sum $\Sigma_z^{ixsyp}R(icxsypz)$ defines a function mapping each customer c to the total duration (sum) of the calls made by that customer;

We introduce the following abbreviations of operators which can be obtained as fractions of the initial ones, *percentage*, *Perc*, and *average*, *Avg*, obtained as a fraction of two counts and/or sums.

Perc $Perc_{x,p}R(icxsypz) = \frac{C^{icsyz}R(icxsypz)}{C^{icsypz}R(icxsypz)}$ defines a function mapping each pair of day/plan (x,p) to the percentage of calls made with plan p in day x with respect to the total number of calls made in day x;

Avg $Avg_z^{ixsyp}R(icxsypz) = \frac{\Sigma_z^{ixsyp}R(icxsypz)}{C^{ixsypz}R(icxsypz)}$ defines a function mapping each customer c to the average duration of calls made by that customer.

The variable y is called the *aggregated variable*, $\bar{x}$ the *argument* of the function, and $\bar{z}$ the *multiset parameter*. The following expressions (of the form $f(\bar{x})$ where f is a function and $\bar{x}$ is an argument): $C^{\bar{z}}\psi(\bar{x},\bar{z})(\bar{x})$, $\Sigma_y^{\bar{z}}\psi(y,\bar{x},\bar{z})(\bar{x})$, and $\Pi_y^{\bar{z}}\psi(y,\bar{x},\bar{z})(\bar{x})$ are *(aggregate) terms*. The free variables of an aggregate term are the free variables of ψ which are neither aggregated nor used as multiset parameter. The *arguments* of the function correspond to the free variables of the aggregate term. For readability, we use the overlined notation

$$\overline{C^{\bar{z}}\psi(\bar{x},\bar{z})}(\bar{x})$$

to distinguish between the variables of ψ and the arguments of the function.

The previous setting allows to express complex functions with nested aggregation and arithmetic. In the sequel though, we consider only *queries* of a very simple form, expressible by one aggregation over a single relation, while more complex functions are necessary for intermediate steps of the rewriting algorithms. We have seen that such queries are relevant for a wide variety of practical situations (e.g. Data Warehousing, Data Cubes, Statistical DBs, etc.) and have been extensively studied [Mal93, GBLP96, HRU96, AAD+96, FSGM+98].

We introduce a convenient shorthand notation for aggregate formulae. We denote by Y, A, B, D, N, respectively the set containing the aggregated variable y, and the sets containing the variables in $\bar{x}^A$, $\bar{x}^B$, $\bar{x}^D$, $\bar{x}^N$, all corresponding implicitly to all the attributes of a relation R, denoted by V. Note that in each case the set of all variables V is partitioned into different subsets.

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

$C(A)$ for $\overline{C^{\bar{x}^N(R)}}(\bar{x}^A)$ $(A \cup N = V)$

$S_Y(A)$ for $\overline{\Sigma_y^{\bar{x}^N(R)}}(\bar{x}^A)$ $(A \cup Y \cup N = V)$

$M_Y(A)$ for $\overline{\Pi_y^{\bar{x}^N(R)}}(\bar{x}^A)$ $(A \cup Y \cup N = V)$

$Perc_{A,B} = \frac{C(A,B)}{C(A)}$ for $\frac{\overline{C^{\bar{x}^N(R)}}(\bar{x}^A,\bar{x}^B)}{\overline{C^{\bar{x}^D(R)}}(\bar{x}^A)}$ $(N \cup A \cup B = D \cup A = V)$

$Avg_Y(A) = \frac{S_Y(A)}{C(A)}$ for $\frac{\overline{\Sigma_y^{\bar{x}^N(R)}}(\bar{x}^A)}{\overline{C^{y\bar{x}^N}R}(\bar{x}^A)}$ $(A \cup Y \cup N = V)$

We also use this notation to manipulate functions. F denotes a function symbol.

$\overline{\Sigma^N(F(N,A))}(A)$ for $\overline{\Sigma_s^{\bar{x}^N(s=F(\bar{x}^N,\bar{x}^A))}}(\bar{x}^A)$

$\overline{\Sigma^{YN}(Y*F(Y,N,A))}(A)$ for $\overline{\Sigma_s^{\bar{y}\bar{x}^N(s=y*F(y,\bar{x}^N,\bar{x}^A))}}(\bar{x}^A)$

In the sequel, we denote queries and views by the corresponding aggregate function, e.g. we say that the query $S_Y(A)$ can be rewritten by using the view $C(Y, A, B)$.

The query and views of Example 1 are defined by

Q1: $S_{Qty}(Product, Color)$

V1: $S_{Qty}(Product, Color, Date)$

V2: $S_{Qty}(Product, State, Date)$

V3: $S_{Qty}(Color, State, Date)$

while the queries and views of Example 2 are defined by:

V1: $\frac{C(Day,Source,Dest,Plan)}{C(Day,Source)}$

Q1: $\frac{C(Day,Source,Plan)}{C(Day,Source)}$

Q2: $\frac{C(Day,Source,Dest,Plan)}{C(Day)}$

Q3: $\frac{C(Day,Source,Dest,Plan)}{C(Day,Source,Plan)}$

V2: $\frac{C(Day,Source,Dur)}{C(Day)}$

6. COMPLETE REWRITING

In this section, we present a first algorithm, **AggRew**, to rewrite queries using count, sum and multiply views. More precisely, the algorithm takes as input a query $F(X)$ of the form $C(X)$, $S_Y(X)$ or $M_Y(X)$, $Perc_{X,X'}$, or Avg_Y, to be rewritten by using views $f_i(W_i)$ of the form $C(X)$, $S_Y(X)$ or $M_Y(X)$ only.

It is easily shown that **AggRew** correctly computes the query of Example 1 by using the view $V1$ and that it does not compute any rewriting using (only) the views $V2$ and $V3$. The algorithm also computes the queries $Q1$, $Q2$ and $Q3$ of Example 2, by using the view $V1$ stored in flat form (in practice $C(Day, Source, Dest, Plan)$ is stored).

The complexity of the algorithm **AggRew** is rather low as shown by the next result.

The algorithm AggRew

Input: a query $F(X)$ to rewrite ($F = C$, S_Y, M_Y, $Perc_{X,X'}$, Avg_Y) and a collection of views $f_i(W_i)$ ($f_i = C$,S_{U_i},M_{U_i})

Output: a rewritten query $F'(X)$ using only the views or a failure message

switch(F)
case C:
  for each count view $C(W_i)$
    if $X \subseteq W_i$ **then set** $F'(X) = \overline{\Sigma^{W_i - X}(C(W_i))}(X)$ and **exit**
case S_Y:
  for each count view $C(W_i)$
    if $(X \cup Y) \subseteq W_i$ **then set** $F'(X) = \overline{\Sigma^{W_i - X}(Y * C(W_i))}(X)$ and **exit**
  for each sum view $S_{U_i}(W_i)$
    if $Y = U_i$ and $X \subseteq W_i$ **then set** $F'(X) = \overline{\Sigma^{W_i - X}(S_{U_i}(W_i))}(X)$ and **exit**
case M_Y:
  for each count view $C(W_i)$
    if $(X \cup Y) \subseteq W_i$ **then set** $F'(X) = \overline{\Pi^{W_i - X}(Y^{C(W_i)})}(X)$ and **exit**
  for each multiply view $M_{U_i}(W_i)$
    if $Y = U_i$ and $X \subseteq W_i$ **then set** $F'(X) = \overline{\Pi^{W_i - X}(M_{U_i}(W_i))}(X)$ and **exit**
case $Perc_{A,B}$
  /* $F(X) = \frac{C(A,B)}{C(A)}$ $(X = A \cup B)$ */
  for each count view $C(W_i)$
    if $(A \cup B) \subseteq W_i$ **then set** $F'(X) = \frac{\overline{\Sigma^{W_i - (A \cup B)}(C(W_i))}(A,B)}{\overline{\Sigma^{W_i - A}(C(W_i))}(A)}$ and **exit**
case Avg_Y:
  for each count view $C(W_i)$
    if $(X \cup Y) \subseteq W_i$ **then set** $F'(X) = \frac{\overline{\Sigma^{W_i - X}(Y * C(W_i))}(X)}{\overline{\Sigma^{W_i - X}(C(W_i))}(X)}$ and **exit**
  for each sum view $S_{U_j}(W_j)$
    if $Y = U_j$ and $X \subseteq W_j$
    then for each count view $C(W_i)$
      if $X \subseteq W_i$ **then set** $F'(X) = \frac{\overline{\Sigma^{W_j - X}(S_{U_j}(W_j))}(X)}{\overline{\Sigma^{W_i - X}(C(W_i))}(X)}$ and **exit**
    exit with a failure message
endswitch
/*all attempts to rewrite have failed*/
exit with a failure message

PROPOSITION 3. *The algorithm* **AggRew** *terminates in time linear in the number of views and their arities.*

Proof. Each view function is examined at most once by the algorithm, and the variables of each function are scanned once to check set containment. Note that the inner loop that scans the count views does not produce (together with the outer loop that scans the sum views) a quadratic time: the count views are scanned only once after a "good" sum view has been detected. If the scan does not detect also

a "good" count view, the algorithm exits with a failure message. ∎

The main result of this section is that the algorithm **AggRew** is sound and complete.

THEOREM 4. *The algorithm* **AggRew** *is sound and complete.*

The proof of Theorem 4 relies on a series of lemmas for the completeness part.

Proof of Theorem 4. The soundness is rather straightforward. It suffices to verify the following equations.

count $C(X) = \overline{\Sigma^{W_i - X}(C(W_i))}(X)$ if $X \subseteq W_i$

sum $S_Y(X) = \overline{\Sigma^{W_i - X}(Y * C(W_i))}(X)$ if $(X \cup Y) \subseteq W_i$
$S_Y(X) = \overline{\Sigma^{W_i - X}(S_{U_i}(W_i))}(X)$ if $Y = U_i$ and $X \subseteq W_i$

multiply $M_Y(X) = \overline{\Pi^{W_i - X}(Y^{C(W_i)})}(X)$ if $(X \cup Y) \subseteq W_i$
$M_Y(X) = \overline{\Pi^{W_i - X}(M_{U_i}(W_i))}(X)$ if $Y = U_i$ and $X \subseteq W_i$

percentage $Perc_{A,B}(X) = \frac{\overline{\Sigma^{W_i - (A \cup B)}(C(W_i))}(A,B)}{\overline{\Sigma^{W_i - A}(C(W_i))}(A)}$ if $(A \cup B) \subseteq W_i$

average $Avg_Y(X) = \frac{\overline{\Sigma^{W_i - X}(Y * C(W_i))}(X)}{\overline{\Sigma^{W_i - X}(C(W_i))}(X)}$ if $(X \cup Y) \subseteq W_i$
$Avg_Y(X) = \frac{\overline{\Sigma^{W_j - X}(S_{U_j}(W_j))}(X)}{\overline{\Sigma^{W_i - X}(C(W_i))}(X)}$ if $Y = U_j$ and $X \subseteq W_j$ and $X \subseteq W_i$

The completeness follows from a series of lemmas. The first lemma is devoted to the rewriting of count queries using count, sum and multiply views. It shows that sum and multiply views are useless and that summarization is the only operation that allows to obtain a count query from a collection of count views. The second and third lemmas show that only summarization and a particular form of aggregation based on count views can be used to rewrite sum and multiply queries. Finally, the last two lemmas are devoted to the case of average and percentage. ∎

LEMMA 5. *A collection of count, sum and multiply views $f_i(W_i)$ subsumes a count query $C(X)$ iff at least one of the f_i is of the form $C(X, Z')$ (Z' possibly empty).*

LEMMA 6. *A collection of count, sum and multiply views $f_i(W_i)$ subsumes a sum query $S_Y(X)$ iff at least one of the f_i is of the form $C(Y, X, Z')$ or of the form $S_Y(X, Z')$ (Z' possibly empty).*

LEMMA 7. *A collection of count, sum and multiply views $f_i(W_i)$ subsumes a multiply query $M_Y(X)$ iff at least one of the f_i is of the form $C(Y, X, Z')$ or of the form $M_Y(X, Z')$ (Z' possibly empty).*

The proof of these lemmas involve combinatorial techniques. The main difficulty relies in the construction of two instances such that the query returns distinct values on the two instances, while all views that are not of the form described in the lemma return identical values on both instances, thus showing that they do not subsume the query. We give a full proof of Lemma 5 and Lemma 6 (The proof of Lemma 7 is analogous to that of Lemma 6).

Proof of Lemma 5. The $\Leftarrow$ part of the proof is easy. Since the query is a summarized version of one of the views:

$$C(X) = \overline{\Sigma^{Z'}(C(X,Z'))}(X)$$

the collection of views certainly subsumes the query.

For the $\Rightarrow$ part it is easily shown that any view (and hence, by Proposition 1 any collection of views) that does not satisfy the conditions of the lemma is subsumed by the following collection of views: $C(X_{-i}, Z)$, $S_{X_i}(X_{-i}, Z)$, $S_{Z_j}(X, Z_{-j})$, $M_{X_i}(X_{-i}, Z)$, $M_{Z_j}(X, Z_{-j})$ (where $i = 1, \ldots, |X| = N$, $j = 1, \ldots, |Z| = M$ and by X_{-i} we denote the set X without the i-th variable).

Indeed, any such view can be computed by summarizing one of the above mentioned views, e.g.

$C(X_1, Z) = \overline{\Sigma^{X-\{X_1,X_N\}}(C(X_{-N},Z))}(X_1, Z)$,
$S_{Z_1}(X, Z_M) = \overline{\Sigma^{Z-\{Z_1,Z_M\}}(S_{Z_1}(X,Z_{-1}))}(X, Z_M)$,
etc.

Consequently, if these views are unable to distinguish two instances, then any other collection of views not satisfying the condition of the lemma is unable to distinguish them.

We consider a couple of instances I_1 and I_2 and show that the view functions return the same values on both instances, while the query function returns distinct values on the two instances. The instances are based on a *chessboard* that is first illustrated in the case where $N = M = 1$.

For each variable in X we consider a domain of 6 values $d_1, d_2, d_3, d_4, d_5, d_6$ such that the equalities $d_1 + d_3 + d_5 = d_2 + d_4 + d_6$ and $d_1 * d_3 * d_5 = d_2 * d_4 * d_6$ hold (e.g. $d_1 = 16$, $d_2 = 10$, $d_3 = 15$, $d_4 = 20$, $d_5 = 5$ and $d_6 = 6$). For each variable in Z we similarly consider a domain of 3 values c_1, c_2, c_3 such that the equalities $c_1 + c_3 = c_2$ and $c_1 * c_3 = c_2$ hold (e.g. $c_1 = 1.5$, $c_2 = 4.5$ and $c_3 = 3$).

The case where $N = M = 1$ is illustrated in Figure 1. White little squares represent tuples of I_1, while gray little squares represent tuples of I_2.

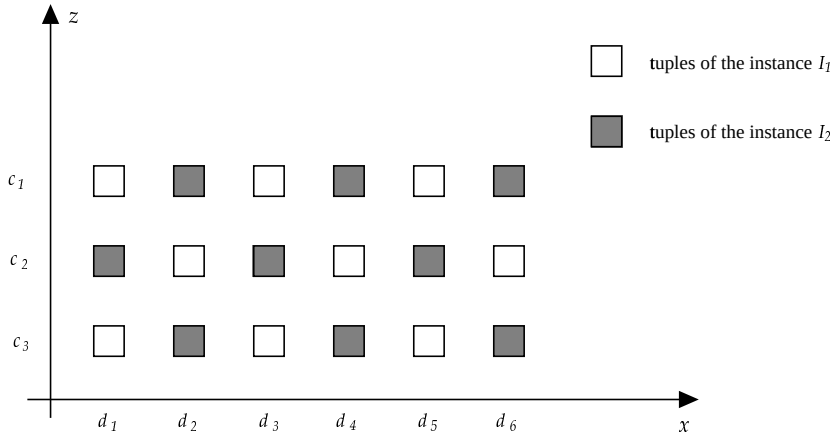

FIG. 1 The pair of chessboard instances

It is easily shown that the query function $C(X)$ returns distinct values on the two instances, namely $C(d_1) = 2$ on I_1 and $C(d_1) = 1$ on I_2.

On the other hand all functions not satisfying the conditions of the Proposition are unable to distinguish the two instances. On both instances we have indeed:

- $\forall Z\ C(Z) = 3$
- $\forall Z\ S_X(Z) = 36$
- $\forall X\ S_Z(X) = 4.5$
- $\forall Z\ M_X(Z) = 1200$
- $\forall X\ M_Z(X) = 4.5$

The chessboard instantiation outlined above can be generalized as follows.

We consider the set of all tuples that can be built by using the 6 values d_i for the variables in X and the 3 values c_j for the variables in Z. Then such set of tuples is partitioned into the two instances I_1 and I_2 as follows.

Given the tuple $t_0 = \langle d_1, \ldots, d_1, a_1, c_1, \ldots, c_1 \rangle$ we consider the *distance* of any other tuple from it, defined as the sum of all constant indexes minus $(N+M)$. The instance I_1 is formed by all tuples having an even or null distance from t_0, I_2 by all tuples having an odd difference.

It can be shown that the query function $C(X)$ returns distinct values on the two instances, e.g. $C(d_1, \ldots, d_1) = \left\lceil \frac{3^M}{2} \right\rceil$ on I_1, while $C(d_1, \ldots, d_1) = \left\lfloor \frac{3^M}{2} \right\rfloor$ on I_2

On the other hand the view functions are unable to distinguish the two instances. On both instances we have indeed (we assume the constants have the example values indicated above):

- $\forall X_{-i}, Z\ C(X_{-i}, Z) = 3$
- $\forall X_{-i}, Z\ S_X(X_{-i}, Z) = 36$
- $\forall X, Z_{-j}\ S_Z(X, Z_{-j}) = 4.5$
- $\forall X_{-i}, Z\ M_X(X_{-i}, Z) = 1200$
- $\forall X, Z_{-j}\ M_Z(X, Z_{-j}) = 4.5$

Thus showing that the considered collection of views (and hence any other collection of views that does not satisfy the conditions of the Lemma) does not subsume the indicated query. ∎

Proof of Lemma 6. The $\Leftarrow$ part of the proof is easy and based on the following identities:

$$S_Y^Z(X) = \overline{\Sigma^{Z'}(S_Y^{Z''}(X,Z'))}(X)$$

$$S_Y^Z(X) = \overline{\Sigma^{Y,Z'}(Y * C(X,Y,Z'))}(X)$$

For the $\Rightarrow$ part it is easily shown that any view (and hence, by Proposition 1 any collection of views) that does not satisfy the conditions of the lemma is subsumed by the following collection of views: $C(X_{-i}, Y, Z)$, $C(X, Z)$, $S_{X_i}(X_{-i}, Y, Z)$, $S_Y(X_{-i}, Z)$, $S_{Z_j}(X, Y, Z_{-j})$, $M_{X_i}(X_{-i}, Y, Z)$, $M_Y(X_{-i}, Z)$, $M_{Z_j}(X, Y, Z_{-j})$ (where $i = 1, \ldots, |X| = N$, $j = 1, \ldots, |Z| = M$ and by X_{-i} we denote the set X without the i-th variable).

Indeed, any such view can be computed by summarizing one of the above mentioned views (see the proof of the previous lemma). Consequently, if these functions are unable to distinguish two instances, then any other collection of views not satisfying the condition of the lemma is unable to distinguish them.

As in the proof of Lemma 5 we consider a couple of instances I_1, I_2 and show that the view functions return the same values, while the query return distinct values on the two instances. The instances are based on a *chessboard* that is illustrated in the simple case where $N = M = 1$ (see Figure 2: the white little cubes represent the tuples of I_1, the gray ones the tuples of I_2).

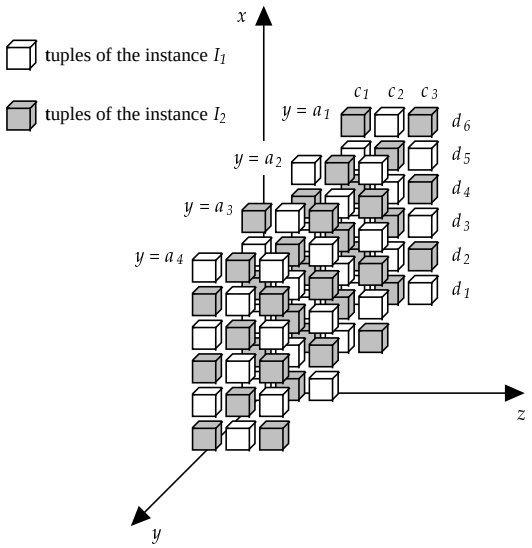

FIG. 2 The pair of chessboard instances

For each variable in X we consider a domain of 6 values $d_1, d_2, d_3, d_4, d_5, d_6$ such that the equalities $d_1 + d_3 + d_5 = d_2 + d_4 + d_6$ and $d_1 * d_3 * d_5 = d_2 * d_4 * d_6$ hold (e.g. $d_1 = 16$, $d_2 = 10$, $d_3 = 15$, $d_4 = 20$, $d_5 = 5$ and $d_6 = 6$). For each variable in Z we similarly consider a domain of 3 values c_1, c_2, c_3 such that the equalities $c_1 + c_3 = c_2$ and $c_1 * c_3 = c_2$ hold (e.g. $c_1 = 1.5$, $c_2 = 4.5$ and $c_3 = 3$). Finally, we consider a domain of four values $\{a_1, a_2, a_3, a_4\}$ for the variable in Y, satisfying the condition $a_1 * a_3 = a_2 * a_4$ (e.g. $a_1 = 1$, $a_2 = 2$, $a_3 = 6$ and $a_4 = 3$).

It is easily shown that the query function $S_Y(X)$ returns distinct values on the two instances, e.g. $S_Y(d_1) = S_Y(16) = 2a_1 + a_2 + 2a_3 + a_4 = 19$ on I_1 and $S_Y(d_1) = S_Y(16) = a_1 + 2a_2 + a_3 + 2a_4 = 17$ on I_2. On the other hand all functions not satisfying the conditions of the lemma are unable to distinguish the two instances and produce the same values on both instances. On both instances we have indeed:

- $\forall Y, Z\ C(Y, Z) = 3$
- $\forall X, Z\ C(X, Z) = 2$
- $\forall Y, Z\ S_X(Y, Z) = 36$

- $\forall Z\ S_Y(Z) = 3(a_1 + a_2 + a_3 + a_4) = 36$
- $\forall X, Y\ S_Z(X, Y) = 4.5$
- $\forall Y, Z\ M_X(Y, Z) = 1200$
- $\forall X, Z\ M_Y(X, Z) = a_1 a_3 = a_2 a_4 = 6$
- $\forall X, Y\ M_Z(X, Y) = 4.5$

The chessboard instantiation outlined above can be generalized as follows.

We consider the set of all tuples that can be built by using the 6 values d_i for the variables in X the 3 values c_j for the variables in Z and the four values a_h for the variable in Y. Then such set of tuples is partitioned into the two instances I_1 and I_2 as follows.

Given the tuple $t_0 = \langle d_1, \ldots, d_1, a_1, c_1, \ldots, c_1 \rangle$ we consider the *distance* of any other tuple from it, defined as the sum of all constant indexes minus $(N+M)$. The instance I_1 is formed by all tuples having an even or null distance from t_0, I_2 by all tuples having an odd difference.

It can be shown that the query function $C(X)$ returns distinct values on the two instances, e.g. $C(d_1, \ldots, d_1) = \left\lceil \frac{3^M}{2} \right\rceil (a_1 + a_3) + \left\lfloor \frac{3^M}{2} \right\rfloor (a_2 + a_4)$ on I_1, while $C(d_1, \ldots, d_1) = \left\lfloor \frac{3^M}{2} \right\rfloor (a_1 + a_3) + \left\lceil \frac{3^M}{2} \right\rceil (a_2 + a_4)$ on I_2

On the other hand the view functions are unable to distinguish the two instances. On both instances we have indeed (we assume the constants have the example values indicated above):

- $\forall X_{-i}, Y, Z\ C(X_{-i}, Y, Z) = 3$
- $\forall X, Z\ C(X, Z) = 2$
- $\forall X_{-i}, Y, Z\ S_{X_i}(X_{-i}, Y, Z) = 36$
- $\forall X_{-i}, Z\ S_Y(X_{-i}, Z) = 3(a_1 + a_2 + a_3 + a_4) = 36$
- $\forall X, Y, Z_{-j}\ S_{Z_j}(X, Y, Z_{-j}) = 4.5$
- $\forall X_{-i}, Y, Z\ M_{X_i}(X_{-i}, Y, Z) = 1200$
- $\forall X_{-i}, Z\ M_Y(X_{-i}, Z) = a_1 a_3 = a_2 a_4 = 6$
- $\forall X, Y, Z_{-j}\ M_{Z_j}(X, Y, Z_{-j}) = 4.5$

Thus showing that the considered collection of views (and hence any other collection of views that does not satisfy the conditions of the Lemma) does not subsume the indicated query. ∎

The two following lemmas are devoted to the average and the percentage.

LEMMA 8. *A collection of count, sum and multiply views $f_i(W_i)$ subsumes a percentage query $F(X) = F(A, B) = \frac{C(A,B)}{C(A)}$ iff at least one of f_i is of the form $C(X, Z')$ (Z' possibly empty).*

Proof. The $\Leftarrow$ part of the proof is obvious. For the converse, suppose by contradiction that none of the f_i's is of the form $C(X, Z') = C(A, B, Z')$, but the views subsume the query. Let us add the view $C(A)$ to the collection of views (if it is not already one of them). None of the views f_i, $C(A)$ is of the form $C(A, B, Z')$. However, since $C(A, B) = F(A, B) \cdot C(A)$ and the f_i's subsume $F(A, B)$, C(A,B) is subsumed by $\{f_i\} \cup C(A)$, which contradicts Lemma 5. ∎

LEMMA 9. *A collection of count, sum and multiply views $f_i(W_i)$ subsumes an average query $F(A) = \frac{S_Y(A)}{C(A)}$ iff at least one of the f_i's subsumes $S_Y(A)$ and at least one of f_i's subsumes $C(A)$.*

Proof. The $\Leftarrow$ part of the proof is obvious. For the converse, suppose by contradiction that none of the f_i's subsumes $S_Y(A)$, but the views subsume the query. Let us add the view $C(A)$ to the collection of views (if it is not already one of them). None of the views f_i, $C(A)$ subsumes $S_Y(A)$. However, since $S_Y(A) = F(A) \cdot C(A)$ and the f_i's subsume $F(A)$, $S_Y(A)$ is subsumed by $\{f_i\} \cup C(A)$, which leads to a contradiction. The proof that at least one of the f_i must subsume $C(A)$ is analogous. ∎

7. MATERIALIZED RATIO VIEWS

In this section, we consider the case were ratio views can be materialized. Two distinct strategies can be pursued: (i) only materialization of views with flat aggregation (e.g. count, sum and multiply), and computation of complex aggregation (e.g. average, percentage) from the former, as was done in the previous section, and (ii) materialization of any aggregate views. While the choice might not be visible to the user who might see logical views, explicitly materialized or derived from the materialized ones, it has a serious impact on the queries which can be answered, the complexity of query rewriting, and the incremental maintenance of views after updates.

In the sequel, we consider an algorithm for rewriting queries and views with ratio aggregation. As will be seen we obtain a completeness result which is weaker than in the previous case. For simplicity, we concentrate on the fragment of the language with only count and sum aggregation, but no product. It is easy to extend the results to the product since it has little interaction with percentages and averages, but the rules would be tedious to write.

The following theorems show limitations on the derivability of ratio views.

THEOREM 10. *A percentage view $v : \frac{C(N_v)}{C(D_v)}$ subsumes a percentage query $q : \frac{C(N_q)}{C(D_q)}$ iff $D_v \subseteq D_q$ and $N_q \subseteq N_v$.*

Proof. The $\Leftarrow$ part is straightforward and is demonstrated later as a consequence of the soundness of the E-rules (Proposition 12). For the $\Rightarrow$ part we demonstrate that if $D_v \not\subseteq D_q$ or $N_q \not\subseteq N_v$ then the view cannot subsume the query. Particularly, if $N_q \not\subseteq N_v$, then the view does not subsume the query even if stored in flat form, by Lemma 8. Hence, assume that $D_v \not\subseteq D_q$, but $N_q \subseteq N_v$. It follows that there exists at least one variable η which is in D_v (and consequently also in N_v) but not in D_q. Let us consider also one variable ξ which is not in N_v (such a variable certainly exists because the count removes at least one variable). For simplicity, let us assume that ξ and η correspond to the last two attributes of R and let us

consider a tuple value $\bar{c}$ for the other attributes, two values $\hat{\xi}_1$ and $\hat{\xi}_2$ for ξ, and three values $\hat{\eta}_1$, $\hat{\eta}_2$, $\hat{\eta}_3$ for the variable η. Finally, let us consider the two instances

$I_1 = \{\langle \bar{c}, \hat{\xi}_1, \hat{\eta}_1 \rangle, \langle \bar{c}, \hat{\xi}_2, \hat{\eta}_2 \rangle\}$, and

$I_2 = \{\langle \bar{c}, \hat{\xi}_2, \hat{\eta}_1 \rangle, \langle \bar{c}, \hat{\xi}_1, \hat{\eta}_2 \rangle, \langle \bar{c}, \hat{\xi}_2, \hat{\eta}_3 \rangle\}$.

It is easily verified that v has uniformly value 1 both for I_1 and I_2, while q has uniformly value $1/2$ for I_1 and value $1/3$ for I_2. Consequently $v \not\models q$. ∎

The previous theorem shows in particular that the query $Q2$ of Example 2 cannot be rewritten by using only view $V1$.

THEOREM 11. *An average view* $v : \frac{S_Y(A)}{C(A)}$ *subsumes an average query* $q : \frac{S_Y(B)}{C(B)}$ *iff* $A = B$.

The proof is in the same spirit as that of Theorem 10.

Proof. The $\Leftarrow$ part is obvious. For the $\Rightarrow$ part, we demonstrate that if $B \neq A$, the view does not subsume the query. First, if $B \not\subseteq A$, the view can not subsume the query even if stored in flat form, by Lemma 9. Hence, assume that $B \subset A$. It follows that there exists at least one variable ξ which is in A but not in B. For simplicity, let us assume that ξ and y are the last attributes of R. We consider a tuple value $\bar{c}$ for all attributes but ξ and y, a couple of values $\hat{\xi}_1$ and $\hat{\xi}_2$ for ξ and some numeric values for the attribute y. Finally, let us consider the two instances

$I_1 = \{\langle \bar{c}, \hat{\xi}_1, 1 \rangle, \langle \bar{c}, \hat{\xi}_2, 2 \rangle\}$, and

$I_2 = \{\langle \bar{c}, \hat{\xi}_1, 1 \rangle, \langle \bar{c}, \hat{\xi}_2, 0 \rangle, \langle \bar{c}, \hat{\xi}_2, 4 \rangle\}$.

It is easily verified that $v(I_1) = v(I_2) = \{\langle \bar{c}', \hat{\xi}_1, 1 \rangle, \langle \bar{c}', \hat{\xi}_2, 2 \rangle\}$ (where $\bar{c}'$ is the projection of $\bar{c}$ on the attributes in A), while $q(I_1) = \{\langle \bar{c}'', 3/2 \rangle\}$ and $q(I_2) = \{\langle \bar{c}'', 5/3 \rangle\}$ (where $\bar{c}''$ is the projection of $\bar{c}$ on the attributes in B). Consequently $v \not\models q$. ∎

The rewriting of queries in presence of ratios is much more complex than the rewriting using flat views, because of the intricacy of the arithmetic computation involved, and its alternation with aggregation operations. The algorithm proposed in this section extends the previous ones to the case of views stored as ratios, and in particular allows the rewriting of the queries $Q1$ and $Q3$ of Example 2, by having $V1$ stored as a ratio.

Although the search space dramatically increases with ratios, the results of completeness obtained in the previous section together with Theorems 10 and 11 restrict the sound derivations on percentage and average queries and views. We next introduce a simple and intuitive set of elementary rewriting rules, called, *E-rules*, based on summarizations and reductions of ratios, applied to one or two count, sum, percentage and average functions, which produce one new function of such types.

Note that we consider rewriting systems in which rules have one or several terms in the left part (and in the sequel similarly one or several terms in the right part). A rewriting rule therefore rewrites one or more terms into one or more terms. We will be mainly interested in applying these rules to sets of terms (view functions).

To each rewriting rule $\xi : l \rightarrow r$, we associate a non-deterministic operator Θ_ξ which maps a set of terms (views), $\mathcal{V}$, to $\Theta_\xi(\mathcal{V}) = (\mathcal{V} - l) \cup r$, that is to the set obtained by removing the terms "used" in the left hand side of the rule and by adding the terms "obtained" in the right hand side, much like in proofs of linear logic.

E-rules over ratio views

E1: $C(A,B) \rightarrow C(A)$

E2: $S_Y(A,B) \rightarrow S_Y(A)$

E3: $C(Y,A) \rightarrow S_Y(A)$

E4: $C(A,B) \rightarrow \frac{C(A,B)}{C(A)}$

E5: $S_Y(A), C(A) \rightarrow \frac{S_Y(A)}{C(A)}$

E6: $\frac{C(A,B)}{C(A)}, C(A) \rightarrow C(A,B)$

E7: $\frac{S_Y(A)}{C(A)}, C(A) \rightarrow S_Y(A)$

E8: $\frac{S_Y(A)}{C(A)}, S_Y(A) \rightarrow C(A)$

E9: $\frac{C(Y,A)}{C(A)} \rightarrow \frac{S_Y(A)}{C(A)}$

E10: $\frac{S_Y(A,B)}{C(A,B)}, \frac{C(A,B)}{C(A)} \rightarrow \frac{S_Y(A)}{C(A)}$

E11: $\frac{C(A,B,D)}{C(A,B)}, \frac{C(A,B)}{C(A)} \rightarrow \frac{C(A,B,D)}{C(A)}$

E12: $\frac{C(A,B,D)}{C(A)} \rightarrow \frac{C(A,B)}{C(A)}$

E13: $\frac{C(A,B,D)}{C(A)} \rightarrow \frac{C(A,B,D)}{C(A,B)}$

To a rewriting system, T, we associate a non-deterministic operator Θ_T which maps a set of terms, $\mathcal{V}$, to $\Theta_T(\mathcal{V}) = (\mathcal{V} - l) \cup r$, for some rule $\xi : l \rightarrow r$ in T.

In the sequel, we will be interested in minimal fixpoints of the operator Θ_K associated to K-rules, obtained by iterating from an initial set of views $\mathcal{V}$. We call such sets of views, *fixpoints of* Θ_K *on* $\mathcal{V}$.

A derivation from an initial set of views is a sequence of sets of views:

$$\mathcal{V} \rightarrow_T \mathcal{V}_1 \rightarrow_T \cdots \rightarrow_T \mathcal{V}_n \rightarrow_T \cdots$$

We first prove that the E-rules are sound, that is, they only allow the derivation of views that are subsumed by the initial views, or more precisely, the views on the left part subsume the views on the right part.

PROPOSITION 12. *The E-rules are sound.*

Proof. The proof is based on a simple case analysis:

E1: $C(A) = \overline{\Sigma^B(C(A,B))}(A)$

E2: $S_Y(A) = \overline{\Sigma^B(S_Y(A,B))}(A)$

E3: $S_Y(A) = \overline{\Sigma^Y(Y*C(Y,A))}(A)$

E4: $\frac{C(A,B)}{C(A)} = \frac{C(A,B)}{\overline{\Sigma^B(C(A,B))}(A)}$

E5: it follows from the definition of average

E6: it is a simple reduction of ratios

E7: it is a simple reduction of ratios

E8: it is a simple reduction of ratios

E9: $\frac{S_Y(A)}{C(A)} = \overline{\Sigma^Y(\frac{C(Y,A)}{C(A)})}(A)$

E10: $\frac{S_Y(A)}{C(A)} = \overline{\Sigma^B(\frac{S_Y(A,B)}{C(A,B)} * \frac{C(A,B)}{C(A)})}(A)$

E11: it is a simple reduction of ratios

E12: $\frac{C(A,B)}{C(A)} = \overline{\Sigma^D(\frac{C(A,B,D)}{C(A)})}(A,B)$

E13: $\frac{C(A,B,D)}{C(A,B)} = \frac{\frac{C(A,B,D)}{C(A)}}{\overline{\Sigma^D(\frac{C(A,B,D)}{C(A)})}(A,B)}$

∎

Given a set of views, $\mathcal{V}$, we denote by $\mathcal{V}_E$ the transitive closure of $\mathcal{V}$ under the E-rules. The following simple proposition is fundamental.

PROPOSITION 13. *For a finite set of views $\mathcal{V}$, $\mathcal{V}_E$ is finite, and $\mathcal{V} \approx \mathcal{V}_E$.*

The proof is rather obvious.

We conjecture that E-rules are complete, that is that the transitive closure $\mathcal{V}_E$ of a finite set of views $\mathcal{V}$ contains each view subsumed by $\mathcal{V}$, but the proof cannot be tackled by the same type of combinatorial arguments as in the flat case.

The next proposition considers the size of $\mathcal{V}_E$.

PROPOSITION 14. *The cardinality of the transitive closure, $\mathcal{V}_E$, under the E-rules, of a set of views $\mathcal{V}$ over an initial relation R, is exponential in the arity of the relation R.*

Proof. The number of distinct views definable over R is exponential in the arity of R (for all subsets of attributes of R). On the other hand, some of the E-rules produce such an exponential number of derived views. For example, given a view $C(A)$ the rule E1 produces all the views $C(A')$ for any $A' \subset A$. ∎

Although the previous result might seem harmless, it is in practice inacceptable. Indeed, aggregate views in statistical databases for instance are derived from initial census relations of extremely large arity.

A simple non-deterministic algorithm for rewriting queries using views based on E-rules can be easily designed. The algorithm chooses non-deterministically views and E-rules, and applies them when possible. It can be shown that one branch of the non-deterministic computation computes the rewritten query in a number of steps polynomial in the number of views.

However, the rewriting can be done with a deterministic algorithm, of time complexity polynomial in the number of views and linear in the arity of the initial relation, if a less naive strategy is pursued. In the following we consider a strategy based on a distinct (but equivalent) set of rules. The algorithm is based on the idea of maintaining a minimal number of "kernel" views during the computation, delaying the application of rules that produce an exponential growth.

To achieve this goal, we separate rules that produce views having the same information content as the initial views, called *K-rules*, and rules that strictly decrease the information content of the initial views, called *S-rules*, that are used only at the end of the derivation process to produce the query to be rewritten. Note

K-rules over ratio views

K1: $\frac{S_Y(A)}{C(A)}, S_Y(A,D) \rightarrow C(A), S_Y(A,D)$

K2: $\frac{S_Y(A)}{C(A)}, C(A,D) \rightarrow S_Y(A), C(A,D)$

K3: $\frac{C(A,B)}{C(A)}, C(A,D) \rightarrow C(A,B), C(A,D)$

K4: $\frac{C(Y,A,B)}{C(A)}, S_Y(A,D) \rightarrow C(Y,A,B), S_Y(A,D)$

K5: $\frac{C(A,A',A'',B,B')}{C(A,A')}, \frac{C(A,A',A'',B,B'')}{C(A,A'')} \rightarrow \frac{C(A,A',A'',B,B')}{C(A,A'')}, \frac{C(A,A',A'',B,B')}{C(A,A')}, \frac{C(A,A',A'',B,B'')}{C(A,A'')}$

K6: $\frac{C(A,B,B')}{C(A)}, \frac{S_Y(A,B)}{C(A,B)}, S_Y(A,D) \rightarrow C(A,B,B'), S_Y(A,B), S_Y(A,D)$

K7: $\frac{C(A,B,B')}{C(A)}, \frac{S_Y(A,B)}{C(A,B)}, C(Y,A,D) \rightarrow C(A,B,B'), S_Y(A,B), C(Y,A,D)$

S-rules over ratio views

S1: $C(A,B) \rightarrow C(A)$

S2: $S_Y(A,B) \rightarrow S_Y(A)$

S3: $C(Y,A,B) \rightarrow S_Y(A)$

S4: $C(A,B,D) \rightarrow \frac{C(A,B)}{C(A)}$

S5: $C(Y,A,B) \rightarrow \frac{S_Y(A)}{C(A)}$

S6: $S_Y(A,B), C(A,D) \rightarrow \frac{S_Y(A)}{C(A)}$

S7: $\frac{C(Y,A,D)}{C(A')}$ (where $A' \subseteq A$) $\rightarrow \frac{S_Y(A)}{C(A)}$

S8: $\frac{S_Y(A,B)}{C(A,B)}, \frac{C(A,B,D)}{C(A')}$ (where $A' \subseteq A$) $\rightarrow \frac{S_Y(A)}{C(A)}$

S9: $\frac{C(A,B,D)}{C(A')}$ (where $A' \subseteq A$) $\rightarrow \frac{C(A,B)}{C(A)}$

that some of the S-rules coincide with E-rules, particularly those which produce an exponential growth of the number of views.

We first prove that the rewriting system based on the combination of the K-rules and the S-rules is equivalent to the rewriting system based on the E-rules.

THEOREM 15. *The derivation based on the union of S-rules and K-rules is equivalent to the derivation based on the E-rules.*

Note that it follows from Theorem 15, that K-rules and S-rules are sound.

Proof of Theorem 15. The fact that E-rules are derivable from S-rules and K-rules follows from Lemma 16. The fact that S-rules and K-rules are derivable from E-rules is shown below.

S1: $C(A,B) \overset{(E1)}{\rightarrow} C(A)$

S2: $S_Y(A,B) \overset{(E2)}{\rightarrow} S_Y(A)$

S3: $C(Y,A,B) \overset{(E3)}{\rightarrow} S_Y(A,B) \overset{(E2)}{\rightarrow} S_Y(A)$

S4: $C(A,B,D) \overset{(E1)}{\rightarrow} C(A,B) \overset{(E4)}{\rightarrow} \frac{C(A,B)}{C(A)}$

S5: $C(Y,A,B) \overset{(E3)}{\rightarrow} S_Y(A,B) \overset{(E2)}{\rightarrow} S_Y(A)$

$C(Y,A,B) \overset{(E1)}{\rightarrow} C(A)$

$S_Y(A), C(A) \overset{(E5)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

S6: $S_Y(A,B) \overset{(E2)}{\rightarrow} S_Y(A)$

$C(A,D) \overset{(E1)}{\rightarrow} C(A)$

$S_Y(A), C(A) \overset{(E5)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

S7: $\frac{C(Y,A,D)}{C(A')} \overset{(E13)}{\rightarrow} \frac{C(Y,A,D)}{C(A)} \overset{(E12)}{\rightarrow} \frac{C(Y,A)}{C(A)} \overset{(E9)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

S8: $\frac{C(A,B,D)}{C(A')} \overset{(E13)}{\rightarrow} \frac{C(A,B,D)}{C(A)} \overset{(E12)}{\rightarrow} \frac{C(A,B)}{C(A)}$

$\frac{S_Y(A,B)}{C(A,B)}, \frac{C(A,B)}{C(A)} \overset{(E10)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

S9: $\frac{C(A,B,D)}{C(A')} \overset{(E13)}{\rightarrow} \frac{C(A,B,D)}{C(A)} \overset{(E12)}{\rightarrow} \frac{C(A,B)}{C(A)}$

K1: $S_Y(A,D) \overset{(E2)}{\rightarrow} S_Y(A)$

$\frac{S_Y(A)}{C(A)}, S_Y(A) \overset{(E8)}{\rightarrow} C(A)$

K2: $C(A,D) \overset{(E1)}{\rightarrow} C(A)$

$\frac{S_Y(A)}{C(A)}, C(A) \overset{(E7)}{\rightarrow} S_Y(A)$

K3: $C(A,D) \overset{(E1)}{\rightarrow} C(A)$

$\frac{C(A,B)}{C(A)}, C(A) \overset{(E6)}{\rightarrow} C(A,B)$

K4: $S_Y(A,D) \overset{(E2)}{\rightarrow} S_Y(A)$

$\frac{C(Y,A,B)}{C(A)} \overset{(E12)}{\rightarrow} \frac{C(Y,A)}{C(A)} \overset{(E9)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

$\frac{S_Y(A)}{C(A)}, S_Y(A) \overset{(E8)}{\rightarrow} C(A)$

$\frac{C(Y,A,B)}{C(A)}, C(A) \overset{(E6)}{\rightarrow} C(Y,A,B)$

K5: $\frac{C(A,A',A'',B,B')}{C(A,A')} \overset{(E13)}{\rightarrow} \frac{C(A,A',A'',B,B')}{C(A,A',A'',B)}$

$\frac{C(A,A',A'',B,B'')}{C(A,A'')} \overset{(E12)}{\rightarrow} \frac{C(A,A',A'',B)}{C(A,A'')}$

$\frac{C(A,A',A'',B,B')}{C(A,A',A'',B)}, \frac{C(A,A',A'',B)}{C(A,A'')} \overset{(E11)}{\rightarrow} \frac{C(A,A',A'',B,B')}{C(A,A'')}$

K6: $\frac{C(A,B,B')}{C(A)} \overset{(E12)}{\rightarrow} \frac{C(A,B)}{C(A)}$

$\frac{S_Y(A,B)}{C(A,B)}, \frac{C(A,B)}{C(A)} \overset{(E10)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

$S_Y(A,D) \overset{(E2)}{\rightarrow} S_Y(A)$

$\frac{S_Y(A)}{C(A)}, S_Y(A) \overset{(E8)}{\rightarrow} C(A)$

$\frac{C(A,B,B')}{C(A)}, C(A) \overset{(E6)}{\rightarrow} C(A,B,B')$

$C(A,B,B') \overset{(E1)}{\rightarrow} C(A,B)$

$\frac{S_Y(A,B)}{C(A,B)}, C(A,B) \overset{(E7)}{\rightarrow} S_Y(A,B)$

K7: $\frac{C(A,B,B')}{C(A)} \overset{(E12)}{\rightarrow} \frac{C(A,B)}{C(A)}$

$\frac{S_Y(A,B)}{C(A,B)}, \frac{C(A,B)}{C(A)} \overset{(E10)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

$C(Y,A,D) \overset{(E3)}{\rightarrow} S_Y(A,D)$

$S_Y(A,D) \overset{(E2)}{\rightarrow} S_Y(A)$

$\frac{S_Y(A)}{C(A)}, S_Y(A) \overset{(E8)}{\rightarrow} C(A)$

$\frac{C(A,B,B')}{C(A)}, C(A) \overset{(E6)}{\rightarrow} C(A,B,B')$

$C(A,B,B') \overset{(E1)}{\rightarrow} C(A,B)$

$\frac{S_Y(A,B)}{C(A,B)}, C(A,B) \overset{(E7)}{\rightarrow} S_Y(A,B)$

∎

We now state Lemma 16 which is also fundamental for Theorem 20.

LEMMA 16. *Given a collection of views $\mathcal{V}$, and some fixpoint $\mathcal{V}'$ of Θ_K on $\mathcal{V}$, any view in the closure $\mathcal{V}_E$ under the E-rules, is either (i) directly in $\mathcal{V}'$ or (ii) derivable from $\mathcal{V}'$ by applying exactly one S-rule.*

Proof. The proof is by induction on the structure of the derivation that produces each view in $\mathcal{V}_E$ from the views in $\mathcal{V}$ and is based on a case analysis.
Base case: for each E-rule we assume that the operands are in $\mathcal{V}$ and we demonstrate that the result view is derived by applying the K-rules (if needed) + one of the S-rules:

E1: No K-rule + S1

E2: No K-rule + S2

E3: No K-rule + S3

E4: No K-rule + S4

E5: No K-rule + S5

E6: Apply K3

E7: Apply K2

E8: Apply K1

E9: No K-rule + S6

E10: No K-rule + S7

E11: Apply K5

E12: No K-rule + S8

E13: No K-rule + S9

Inductive step: for each E-rule we assume that the property holds for each operand (i.e. each operand can be obtained by applying the K-rules a finite number of times + one S-rule) and we demonstrate that the property also holds for the resulting view.

In practice for each E-rule we assume (worst case) that all operands are not directly computed by the K-rules, but require an S-rule to be obtained, e.g. if one of the operand is a percentage $\frac{C(A,B)}{C(A)}$ the S-rule required to have it may be S4 or S9 and the views already computed by the K-rules are $C(A,B,D)$ or $\frac{C(A,B,D)}{C(A')}$, respectively.

1. $C(A,B) \overset{(E1)}{\rightarrow} C(A)$ - The operand $C(A,B)$ can only be obtained by S1, i.e. from a view of the form $C(A,B,D)$ (which has been computed by the K-rules by inductive assumption). Since

   $C(A,B,D) \overset{(S1)}{\rightarrow} C(A)$

   the condition is satisfied.

2. $S_Y(A,B) \overset{(E2)}{\rightarrow} S_Y(A)$ - The operands may be obtained by S2 or S3, i.e. either from a view of the form $S_Y(A,B,D)$ or from a view of the form $C(Y,A,B,D)$. Since

   $S_Y(A,B,D) \overset{(S2)}{\rightarrow} S_Y(A)$

   $C(Y,A,B,D) \overset{(S3)}{\rightarrow} S_Y(A)$

   the condition is satisfied.

3. $C(Y,A) \overset{(E3)}{\rightarrow} S_Y(A)$ - The operand can only be obtained by S1. Since

   $C(Y,A,D) \overset{(S3)}{\rightarrow} S_Y(A)$

   the condition is satisfied.

4. $C(A,B) \overset{(E4)}{\rightarrow} \frac{C(A,B)}{C(A)}$ - The operand can only be obtained by S1. Since

   $C(A,B,D) \overset{(S4)}{\rightarrow} \frac{C(A,B)}{C(A)}$

   the condition is satisfied.

5. $S_Y(A), C(A) \overset{(E5)}{\rightarrow} \frac{S_Y(A)}{C(A)}$ - Two ways to obtain the first operand, only one for the second. Since

   $S_Y(A,B), C(A,D) \overset{(S6)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

   $C(Y,A,B)$ ($C(A,D)$ unused) $\overset{(S5)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

   the condition is satisfied.

6. $\frac{C(A,B)}{C(A)}, C(A) \overset{(E6)}{\rightarrow} C(A,B)$ - Two possible subcases:

   $C(A,B,D)$ ($C(A,G)$ unused) $\overset{(S1)}{\rightarrow} C(A,B)$

   $\frac{C(A,B,D)}{C(A')}, C(A,G)$ (where $A' \subseteq A$) $\overset{(K3)}{\rightarrow} C(A,B,D) \overset{(S1)}{\rightarrow} C(A,B)$

7. $\frac{S_Y(A)}{C(A)}, C(A) \overset{(E7)}{\rightarrow} S_Y(A)$ - Four possible subcases

   $C(Y,A,B)$ ($C(A,G)$ unused) $\overset{(S3)}{\rightarrow} S_Y(A)$

   $S_Y(A,B)$ ($C(A,G)$ unused) $\overset{(S2)}{\rightarrow} S_Y(A)$

   $\frac{C(Y,A,D)}{C(A')}, C(A,G) \overset{(K3)}{\rightarrow} C(Y,A,D) \overset{(S3)}{\rightarrow} S_Y(A)$

   $\frac{S_Y(A,B)}{C(A,B)}, \frac{C(A,B,D)}{C(A')}, C(A,G) \overset{(K7)}{\rightarrow} C(A,B,D), S_Y(A,B) \overset{(S2)}{\rightarrow} S_Y(A)$

8. $\frac{S_Y(A)}{C(A)}, S_Y(A) \overset{(E8)}{\rightarrow} C(A)$ - Eight possible subcases

   $C(Y,A,B)$ ($S_Y(A,G)$ unused) $\overset{(S1)}{\rightarrow} C(A)$

   $C(Y,A,B)$ ($C(Y,A,G)$ unused) $\overset{(S1)}{\rightarrow} C(A)$

   $S_Y(A,B), C(A,D)$ ($S_Y(A,G)$ unused) $\overset{(S1)}{\rightarrow} C(A)$

   ($S_Y(A,B), C(A,D)$ unused) $C(Y,A,G) \overset{(S1)}{\rightarrow} C(A)$

   $\frac{C(Y,A,D)}{C(A')}, S_Y(A,G) \overset{(K4)}{\rightarrow} C(Y,A,D) \overset{(S1)}{\rightarrow} C(A)$

   ($\frac{C(Y,A,D)}{C(A')}$ unused) $C(Y,A,G) \overset{(S1)}{\rightarrow} C(A)$

   $\frac{S_Y(A,B)}{C(A,B)}, \frac{C(A,B,D)}{C(A')}, S_Y(A,G) \overset{(K6)}{\rightarrow} S_Y(A,B), C(A,B,D) \overset{(S1)}{\rightarrow} C(A)$

   ($\frac{S_Y(A,B)}{C(A,B)}, \frac{C(A,B,D)}{C(A')}$ unused) $C(Y,A,G) \overset{(S1)}{\rightarrow} C(A)$

9. $\frac{C(Y,A)}{C(A)} \overset{(E9)}{\rightarrow} \frac{S_Y(A)}{C(A)}$ - Two possible subcases

   $C(Y,A,B) \overset{(S5)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

   $\frac{C(Y,A,B)}{C(A')} \overset{(S7)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

10. $\frac{S_Y(A,B)}{C(A,B)}, \frac{C(A,B)}{C(A)} \overset{(E10)}{\rightarrow} \frac{S_Y(A)}{C(A)}$ - Four (double) possible subcases

    $C(Y,A,B)$ (independently of the second operand) $\overset{(S5)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

    $S_Y(A,B), C(A,D)$ (independently of the second operand) $\overset{(S6)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

    $\frac{C(Y,A,D)}{C(A')}$ (independently of the second operand) $\overset{(S7)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

    $\frac{S_Y(A,B)}{C(A,B)}, \frac{C(A,B,D)}{C(A')}$ (independently of the second operand) $\overset{(S8)}{\rightarrow} \frac{S_Y(A)}{C(A)}$

11. $\frac{C(A,B,D)}{C(A,B)}, \frac{C(A,B)}{C(A)} \overset{(E11)}{\rightarrow} \frac{C(A,B,D)}{C(A)}$ - Four possible subcases

    (2) $C(A,B,D)$ (independently of the second operand) $\overset{(S4)}{\rightarrow} \frac{C(A,B,D)}{C(A)}$

    $\frac{C(A,B,D,G)}{C(A',B')}, C(A,B,H) \overset{(K3)}{\rightarrow} C(A,B,D,G) \overset{(S4)}{\rightarrow} \frac{C(A,B,D)}{C(A)}$

    $\frac{C(A,B,D,G)}{C(A',B')}, \frac{C(A,B,H)}{C(A'')} \overset{(K5)}{\rightarrow} \frac{C(A,B,D,G)}{C(A'')} \overset{(S9)}{\rightarrow} \frac{C(A,B,D)}{C(A)}$

12. $\frac{C(A,B,D)}{C(A)} \overset{(E12)}{\rightarrow} \frac{C(A,B)}{C(A)}$ - Two possible subcases

    $C(A,B,D) \overset{(S4)}{\rightarrow} \frac{C(A,B)}{C(A)}$

    $\frac{C(A,B,D,G)}{C(A')} \overset{(S9)}{\rightarrow} \frac{C(A,B)}{C(A)}$

13. $\frac{C(A,B,D)}{C(A)} \overset{(E13)}{\rightarrow} \frac{C(A,B,D)}{C(A,B)}$ - Two possible subcases

    $C(A,B,D) \overset{(S4)}{\rightarrow} \frac{C(A,B,D)}{C(A,B)}$

    $\frac{C(A,B,D,G)}{C(A')} \overset{(S9)}{\rightarrow} \frac{C(A,B,D)}{C(A,B)}$

∎

The next two propositions show that the K-rules are conservative for information content, while the S-rules are strictly loosing information.

PROPOSITION 17. *The views on the right-hand side of each K-rule subsume the views on the left-hand side.*

Proof. It follows from the fact that the views on the left-hand side of each K-rule are derivable (e.g. by using the E-rules) from the views on the right-hand side. ∎

PROPOSITION 18. *The views on the right-hand side of each S-rule do not subsume the views on the left-hand side.*

Proof. For the S-rules S1 - S6, this follows from the fact that even if the views on the right-hand side were stored in flat form they would not subsume the views on the left-hand side, by Lemmas 5 and 6. For S7, we can note that from the view on the left-hand side we can derive $\frac{S_Y(A,D)}{C(A,D)}$ (by using E13 and E9). If the right-hand side view could subsume the left-hand side view, then we would obtain that

$$\frac{S_Y(A)}{C(A)} \models \frac{S_Y(A,D)}{C(A,D)}$$

which is absurd by Theorem 11. Analogously in the case of rules S8 and S9 we would obtain a violation of Theorems 10 and 11, respectively. ∎

We consider now the impact of K-rules on the number of views derived.

PROPOSITION 19. *The cardinality of any fixpoint of Θ_K on $\mathcal{V}$ is at most quadratic in the cardinality of $\mathcal{V}$.*

Proof. First of all note that, apart from K5, the application of the K-rules does not increase the global number of views. Let us consider now the increasing effect of K5. Since the only K-rule that produces new percentage views is K5, it follows that the number of percentage views that can be generated by the iterated application of K5 is bounded by the number of possible combinations of numerators of percentage views with denominators of percentage views, i.e. by the square of the number of percentage views in the database. ∎

The next example shows that a quadratic number can be reached.

EXAMPLE 3. Let us consider a database containing p percentage views

$$\frac{C(x_1,\ldots,x_p,x_{p+1})}{C(x_1)}, \frac{C(x_1,\ldots,x_p,x_{p+2})}{C(x_2)}, \ldots, \frac{C(x_1,\ldots,x_p,x_{2p})}{C(x_p)}$$

The rule K5 produces a quadratic number of distinct percentage functions (given by all possible numerator/denominator combinations). □

We can now introduce the algorithm **AggRew2**. Note that the set $\mathcal{F}$ contains the currently active functions (views), while $\mathcal{N}$ the newly generated ones. Note also that the actual computation of the fixpoint is not required. Only the description of the obtained functions needs to be maintained and the actual computation can be delayed at the end of the rewriting process, when the really necessary functions are computed. Also, the rewriting could be attempted after each K-iteration, without waiting for the completion of the fixpoint. For brevity, only two representative fragments of the fixpoint computation and of the final S-rewriting are shown. The fragments corresponding to the other K- and S-rules are analogous.

The problems of view usability introduced in Section 3 are solved by the proposed algorithm.

EXAMPLE 4. Consider the following queries and view of Example 2:

$$V1 : \frac{C(Day,Source,Dest,Plan)}{C(Day,Source)}$$

$$Q1 : \frac{C(Day,Source,Plan)}{C(Day,Source)}$$

The algorithm AggRew2

Input: a query $F(X)$ to rewrite ($F = C$, S, $Perc$, Avg) and a collection $\mathcal{F}$ of views $f_i(W_i)$ ($f_i = C$, S, $Perc$, Avg)

Output: a rewritten query $F'(X)$ using only the views or a failure message

$\mathcal{N} = \{\}$
/* Compute a fixpoint of Θ_K on $\mathcal{F}$ */
repeat
  $\mathcal{F} = \mathcal{F} \cup \mathcal{N}$
  $\mathcal{N} = \{\}$
  ...
  for each sum view $S_Y(A_s)$ in $\mathcal{F}$
    for each average view $\frac{S_W(A_a)}{C(A_a)}$ in $\mathcal{F}$
      if $Y = W$ **and** $A_a \subseteq A_s$ **then** /* K1 can be applied */
        Compute $C(A_a)$ using K1
        Remove $\frac{S_W(A_a)}{C(A_a)}$ from $\mathcal{F}$
        if $C(A_a) \notin \mathcal{F}$ **then**
          Insert $C(A_a)$ into $\mathcal{N}$
  ...
until $\mathcal{N} = \{\}$
/* Try to rewrite using the S-rules */
switch(F)
...
case Perc: /* the query is of the form $\frac{C(A_Q,B_Q)}{C(A_Q)}$ */
  for each count view $C(A_c)$ in $\mathcal{F}$
    if $A_Q \cup B_Q \subseteq A_c$ **then**
      Compute the rewriting F' of F from $C(A_c)$ by S4
      return F'
  for each percentage view $\frac{C(A_p,B_p)}{C(A_p)}$ in $\mathcal{F}$
    if $A_p \subseteq A_Q$ **and** $A_Q \cup B_Q \subseteq A_p \cup B_p$ **then**
      Compute the rewriting F' of F from $\frac{C(A_p,B_p)}{C(A_p)}$ by S9
      return F'
...
endswitch
/* all attempts to rewrite have failed */
return a failure message

$$Q2 : \frac{C(Day,Source,Dest,Plan)}{C(Day)}$$

$$Q3 : \frac{C(Day,Source,Dest,Plan)}{C(Day,Source,Plan)}$$

$Q1$ and $Q3$ are rewritten by using the view $V1$ and applying the rule S9. Obvi-

ously the query $Q2$ is not computed, since the information content is not sufficient (see Theorem 10). Note that the introduction of one additional view

$$V2 : \frac{C(Day, Source, Dur)}{C(Day)}$$

makes the rewriting possible and indeed the function is computed by the algorithm by applying the rule K5. □

We now state the main result on the proposed algorithm.

THEOREM 20. *The algorithm* **AggRew2** *is sound and complete with respect to the E-rules and its complexity is polynomial in the number of views and linear in the arity of R.*

Proof. The soundness follows from the fact that the derivation based on the union of S-rules and K-rules is equivalent to the derivation based on the E-rules Theorem 15, and the fact that E-rules are sound by Proposition 12. The completeness with respect to the E-rules is a consequence of Lemma 16. The complexity upper-bound follows from Lemma 16 and Proposition 19. At each K-iteration using K1-K4, K6, or K7, at least either one percentage view is replaced by a count view or one average view is replaced by a sum view. A new percentage view is generated by K5. In the first case, it results in an at most linear number of iterations, while in the second case, with K5, at most a quadratic number of iterations can be performed. The S-rules are then applied once only. ∎

Note that if the fixpoint of Θ_K on $\mathcal{F}$ is computed and materialized in advance, a collection of views is obtained, which has the same information content as the initial one and such that the rewriting can be computed in linear time (by using only the second part of **AggRew2**). This is obviously very important, because it shows that even if some ratio views are materialized in the database, the rewriting can still be obtained very efficiently, by a limited expense in terms of storage space. Note also that in practice the quadratic blow up of stored views, produced by the computation of the fixpoint, is very unlikely and there are conversely several cases in which the computation may even decrease the number of views.

8. CONCLUSION

We have considered the general problem of deciding if a query could be rewritten using views, whenever the views contained enough information. We have seen that in theory this is possible, although of little practical interest.

We have then seen that in the case of views and queries with one aggregation over one relation and views stored in flat form, we could obtain a rewriting method which is complete with respect to the information content, and also tractable. This class of queries is of great practical importance in data warehousing and has already been extensively studied [Mal93, GBLP96, HRU96, AAD+96, FSGM+98]. To the best of our knowledge this is the first result of this type.

However, it is not always possible to deal only with flat views. First of all, there are many examples of pulled data that are in ratio format, and they need to be managed. Second, the decision to give access only to ratios might be motivated by privacy reasons. For example, the company of Example 2 would certainly not make available the view containing the detail of the number of calls by day and state,

but would probably exchange some data with other companies on the percentage of calls directed to mobile and wired networks for each day and state.

In the case of materialized ratio views, our result is weaker since the completeness of the algorithm is only shown with respect to a set of rewriting rules. Nevertheless, it is shown that, despite a potentially exponential blow up of the computation, the complexity is tractable and, by a limited expense in terms of storage space, it can even be made linear as in the flat case.

Our results shed lights to the problem of designing materialized views. At first it might seem that it is preferable to *materialize* only flat views, and to offer derived *logical* ratio views such as average. This leads to a complete linear time algorithm for rewriting queries. This approach has been pursued by many authors [GBLP96, Qua96, SDJL96, AAD+96]. However, when the use of materialized ratio views is unavoidable due to pulled data or motivated by privacy issues, the proposed algorithm for ratio views provides a good trade-off between limited storage space overhead and fast responses.

The problem of maintaining materialized views is crucial. The fact that flat views can be maintained while ratio views cannot advocates the use of a design with two levels of views.

Finally, we have focused on equivalent rewriting algorithms. It is clear that there are cases where an equivalent rewriting will not exist, while some data might still be extracted from the views to provide a partial answer to the query using the views using less demanding methods. In the case of aggregate data, partial answers would lead to the use of approximate aggregation functions, with for instance lower and upper bounds instead of exact values. This is a topic of interest for further research. Moreover, the notion of information content that we have proposed captures the exact content, and not a notion of subcontent of the information of the views. This would also require a different concept of information subsumption, which is also of great interest.

ACKNOWLEDGMENTS

The authors wish to thanks the anonymous referees for their suggestions that contributed greatly to improve the paper.

REFERENCES

[AAD+96] S. Agarwal, R. Agrawal, P. Deshpande, A. Gupta, J.F. Naughton, R. Ramakrishnan, and S. Sarawagi. On the computation of multidimensional aggregates. In *Proceedings of the 22th International Conference on Very Large Data Bases, VLDB'96*, pages 506–521, 1996.

[AD98] S. Abiteboul and O.M. Duschka. Complexity of answering queries using materialized views. In *Proc. ACM PODS'98, June 1-3, 1998, Seattle, Washington*, pages 254–263. ACM Press, 1998.

[AGK99] F.N. Afrati, M. Gergatsoulis, and T.G. Kavalieros. Answering queries using materialized views with disjunctions. In *Proc. ICDT'99*, pages 435–452, 1999.

[AGS97] R. Agrawal, A. Gupta, and S. Sarawagi. Modeling multidimensional databases. In *Proceedings of ICDE'97*, pages 232–243. IEEE Computer Society, 1997.

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases.* Addison-Wesley, 1995.

[ALU01] F.N. Afrati, C. Li, and J.D. Ullman. Generating efficient plans for queries using views. In *Proc. SIGMOD 2001.* ACM, 2001.

[BI95] D. Barbará and T. Imielinski. Sleepers and workaholics: Caching strategies in mobile environments. *VLDB Journal*, 4(4):567–602, 1995.

[BPT97] E. Baralis, S. Paraboschi, and E. Teniente. Materialized selection in a multidimensional database. In *Proc. VLDB'97*, pages 156–165, 1997.

[BR99] K.S. Beyer and R. Ramakrishnan. Bottom-up computation of sparse and iceberg cubes. In *Proceedings of ACM SIGMOD International Conference on Management of Data*, pages 359–370, 1999.

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

[CGLV00a] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Answering regular path queries using views. In *Proc. ICDE 2000*, pages 389–398, 2000.

[CGLV00b] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. Query processing using views for regular path queries with inverse. In *Proc. PODS 2000*, pages 58–66, 2000.

[CGLV00c] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. View-based query processing and constraint satisfaction. In *Proc. LICS 2000*, pages 361–371, 2000.

[CGLV00d] D. Calvanese, G. De Giacomo, M. Lenzerini, and M.Y. Vardi. What is view-based query rewriting? In *Proc. KRDB 2000*, pages 17–27, 2000.

[CI98] C.Y. Chan and Y.E. Ioannidis. Bitmap index design and evaluation. In *Proc. SIGMOD'98*, pages 355–366, 1998.

[CKPS95] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *Proc. ICDE'95*, pages 190–200. IEEE Computer Society, 1995.

[CNS99] S. Cohen, W. Nutt, and A. Serebrenik. Rewriting aggregate queries using views. In *Proc. PODS'99*, pages 155–166. ACM Press, 1999.

[CT97] L. Cabibbo and R. Torlone. Querying multidimensional databases. In *Proceedings of DBPL-6*, volume 1369 of *Lecture Notes in Computer Science*, pages 319–335, 1997.

[DG97] O.M. Duschka and M.R. Genesereth. Answering recursive queries using views. In *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, PODS'97*, pages 109–116. ACM Press, 1997.

[DGL00] O.M. Duschka, M.R. Genesereth, and A.Y. Levy. Recursive query plans for data integration. *Journal of Logic Programming*, 43(1):49–73, 2000.

[FSGM+98] M. Fang, N. Shivakumar, H. Garcia-Molina, R. Motwani, and J.D. Ullman. Computing iceberg queries efficiently. In *Proceedings of the 24th International Conference on Very Large Data Bases, VLDB'98*, pages 299–310, 1998.

[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 *Proc. ICDE'96*, pages 152–159, New Orleans, Louisiana USA, February 1996.

[GHRU97] H. Gupta, V. Harinarayan, A. Rajaraman, and J.D. Ullman. Index selection for olap. In *Proceedings of the Thirteenth International Conference on Data Engineering, April 7-11, 1997 Birmingham U.K*, pages 208–219. IEEE Computer Society, 1997.

[GL01] J. Goldstein and P. Larson. Optimizing queries using materialized views: A practical, scalable solution. In *Proc. SIGMOD 2001*. ACM, 2001.

[GM99a] G. Grahne and A.O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *Proc. ICDT'99*, pages 332–347, 1999.

[GM99b] H. Gupta and I.S. Mumick. Selection of views to materialize under a maintenance cost constraint. In *Proc. ICDT'99*, pages 453–470. Springer, 1999.

[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.

[Gup97] H. Gupta. Selection of views to materialize in a data warehouse. In *Database Theory - ICDT'97, 6th International Conference, Delphi, Greece, January 8-10, 1997, Proceedings*, volume 1186, pages 98–112. Springer, 1997.

[Hal00] A.Y. Halevy. Theory of answering queries using views. *Sigmod Record*, 29(4):40–47, Dec. 2000.

[HRU96] V. Harinarayan, A. Rajaraman, and J.D. Ullman. Implementing data cube efficiently. In *Proc. SIGMOD'96*, pages 205–216, Montreal, Canada, 1996.

[JL01] M. Jürgens and H.J. Lenz. Tree based indexes versus bitmap indexes: A performance study. *IJCIS*, 10(3):355–376, 2001.

[LMSS95] A.Y. Levy, A.O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proc. PODS'95*, pages 95–104, 1995.

[LRO96] A.Y. Levy, A. Rajaraman, and J.J. Ordille. Querying heterogeneous information sources using source descriptions. In *Proc. VLDB'96*, pages 251–262. Morgan Kaufmann, 1996.

[LRU96] A.Y. Levy, A. Rajaraman, and J.D. Ullman. Answering queries using limited external processors. In *Proc. PODS'96*, pages 227–237. ACM Press, 1996.

[Mal93] F.M. Malvestuto. A universal-scheme approach to statistical databases containing homogeneous summary tables. *ACM Transactions on Database Systems*, 18(4):678–708, 1993.

[Mit99] P. Mitra. An algorithm for answering queries efficiently using views. Technical report, Stanford University, 1999.

[MM98] F.M. Malvestuto and M. Moscarini. Computational issues connected with the protection of sensitive statistics by auditing sum queries. In *Proc. SSDBM'98*, pages 134–144. IEEE Computer Society, 1998.

[MMR91] F.M. Malvestuto, M. Moscarini, and M. Rafanelli. Suppressing marginal cells to protect sensitive information in a two-dimensional statistical table. In *Proc. PODS'91*, pages 252–258. ACM Press, 1991.

[MQM97] I.S. Mumick, D. Quass, and B.S. Mumick. Maintenance of data cubes and summary tables in a warehouse. In *SIGMOD 1997, Proceedings ACM SIGMOD International Conference on Management of Data, May 13-15, 1997, Tucson, Arizona, USA*, pages 100–111. ACM Press, 1997.

[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, Dec 1987.

[OQ97] P.E. O'Neil and D. Quass. Improved query performance with variant indexes. In *Proc. SIGMOD'97*, pages 38–49, 1997.

[PL00] R. Pottinger and A.Y. Levy. A scalable algorithm for answering queries using views. In *Proc. VLDB 2000*, pages 484–495. Morgan Kaufmann, 2000.

[Qia96] X. Qian. Query folding. In *Proc. ICDE'96*, pages 48–55. IEEE Computer Society, 1996.

[Qua96] D. Quass. Maintenance expressions for views with aggregation. In *VIEW 1996, Proceedings of the Workshop on Materialized Views: Techniques and Applications, June 7, 1996, Monteal, Canada*, pages 110–118, 1996.

[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, Dec 1996.

[RSUV89] R. Ramakrishnan, Y. Sagiv, J.D. Ullman, and M.Y. Vardi. Proof-tree transformation theorems and their applications. In *Proc. PODS'89*, pages 172–181, 1989.

[SDJL96] D. Sristava, S. Dar, H.V. Jagadish, and A.Y. Levy. Answering queries with aggregation using views. In *Proc. VLDB'96*, pages 318–329, 1996.

[Shm87] O. Shmueli. Decidability and expressiveness aspects of logic queries. In *Proc. 6th ACM Symp. on Principles of Database Systems*, pages 237–249, 1987.

[Sho97] A. Shoshani. Olap and statistical databases: Similarities and differences. In *Proc. PODS'97*, pages 183–196, May 1997.

[Wid95] J. Widom. Research problems in data warehousing. In *CIKM 1995, Proceedings of the 4th International Conference on Information and Knowledge Management, November, 1995, Baltimore, Maryland, USA*, pages 25–30, 1995.

[WSD+95] O. Wolfson, A.P. Sistla, S. Dao, K. Narayanan, and R. Raj. View maintenance in mobile computing. *ACM Sigmod Record*, 24(4):22–27, 1995.