A Formal Perspective on the View Selection Problem

Rada Chirkova
Dept. of Computer Science
Stanford University
Stanford, CA 94305
USA
rada@cs.stanford.edu

Alon Y. Halevy
Dept. of Computer Science
University of Washington
Seattle, WA 98195
USA
alon@cs.washington.edu

Dan Suciu
Dept. of Computer Science
University of Washington
Seattle, WA 98195
USA
suciu@cs.washington.edu

Abstract

The view selection problem is to choose a set of views to materialize over a database schema, such that the cost of evaluating a set of workload queries is minimized and such that the views fit into a prespecified storage constraint. The two main applications of the view selection problem are materializing views in a database to speed up query processing, and selecting views to materialize in a data warehouse to answer decision support queries.

We describe several fundamental results concerning the view selection problem. We consider the problem for views and workloads that consist of equality-selection, project and join queries, and show that the complexity of the problem depends crucially on the quality of the estimates that a query optimizer has on the size of the views it is considering to materialize. When a query optimizer has good estimates of the sizes of the views, we show that an optimal choice of views may involve a number of views that is exponential in the size of the database schema. On the other hand, when an optimizer uses standard estimation heuristics, we show that the number of necessary views and the expression size of each view are polynomially bounded.

1 Introduction

The problem of view selection has received significant attention in recent literature [ACN00, CG00, Gup97, GM99, TS97, YKL97, BPT97, GHRU97, HRU96, KM99]. Broadly speaking, the problem is the following: given a database schema $\mathcal{R}$, storage space B and a workload of queries $\mathcal{Q}$, choose a set of views $\mathcal{V}$ over $\mathcal{R}$ to materialize, whose combined size is at most B. The set of views $\mathcal{V}$ is called a *view configuration*. The views can either be materialized within the database, in which case they are available to supplement normal query processing, or they can be materialized in a separate data warehouse, and in that case the workload queries should be answered *only* from the views. The goal of the view selection process is to find a set of views that minimizes the expected cost of evaluating the queries in $\mathcal{Q}$ in either of the above contexts. In addition, the view selection problem may involve choosing a set of indexes on the views, and may consider the cost of updates to the views $\mathcal{V}$.

The original motivation for the view selection problem comes from data warehouse design, where we need to decide which views to store in the warehouse to obtain optimal performance [TS97, BPT97, HRU96]. Another motivation is provided by recent versions of several commercial database systems which support incremental updates of materialized views and now use materialized views to speed up query evaluation. Therefore, choosing an appropriate set of views to materialize in the database is crucial in order to obtain performance benefits from these new features [ACN00].

The view selection problem and its generalizations will play an even greater role in contexts where data needs to be placed intelligently over a wide area network. In these contexts, users are spread over a network, and each location may have different types of query characteristics and/or performance requirements. A very simple version of this problem was considered in the context of data placement in distributed databases(see [Kos00] and, more recently, has bee identified as a key problem in peer to peer computing [GHI+01]. For example, consider the context of ubiquitous computing, where data is both integrated and accessed from many devices (desktops, laptops, PDAs, cellphones) [ILM+00]. Each of these devices has a local store but can also retrieve data at different rates from various points on the network. A key factor in ensuring good performance in such a context is intelligent placement and replication of data at different nodes on the network, which is akin to selecting a set of views at each node. Note that the view selection problem

Proceedings of the 27th VLDB Conference, Roma, Italy, 2001

can be viewed as a special case of this intelligent data placement problem, in which there are only two nodes in the network (the database and the warehouse).

This paper considers the fundamental properties of the view selection problem for workload and view configurations involving conjunctive queries (i.e., queries allowing join, projection and equality selection). Several algorithms have been proposed in the past for solving the view selection problem for such queries (e.g., [TS97, ACN00]), but they all made certain critical tacit assumptions. The first assumption is that the only views that need to be considered for an optimal view configuration are those that are subexpressions of queries in the workload (i.e., that contain a subset of the tables with a subset of the join predicates in the query). The second assumption is that there is some relatively low upper bound on the number of views in an optimal view configuration.

The following example, adapted from [CG00], shows that the first assumption does not hold. We will show later in the paper that the second assumption doesn't hold either.

Example 1.1 Consider a shipping company that serves a number of cities, with fixed delivery schedules between pairs of cities. The company has a centralized database, with a base table T (*source*, d, *dest*) that stores all pairs of cities *source* and *dest*, such that there is a scheduled delivery from *source* to *dest* on day d of the week (a number between 1 and 7).

Agents in each city try to contract shipments to independent truck drivers, by luring them with tours connecting two or more cities. The company predefines a number of tour types to offer to the truck drivers, and agents need to query the database and find out whether the tour requested by the driver exists starting at a given city. Every tour type starts and ends in the *same* city. The simplest tour is the "one-city two-days roundtrip", for which we give the definitions in both SQL and datalog. (We are only concerned with set semantics in this paper, hence the `distinct` in the SQL statement. The equivalent datalog formulation should be self-explanatory.)

```
select distinct T1.source
from T AS T1, T AS T2
where T1.source = T2.dest
and T1.dest = T2.source
and T1.day+1 = T2.day;
```

$$Q_1(X_1) :- T(X_1, D, X_2), T(X_2, D+1, X_1).$$

A more complex tour is the "five cities in five days with a break" tour below:

$$Q_2(X_1) :- T(X_1, 1, X_2), T(X_2, 1, X_3), T(X_3, 2, X_4), T(X_4, 2, X_5), T(X_5, 4, X_4), T(X_4, 5, X_1)$$

Here the break is on Wednesday (day #3).

Assume that the company predefines a few dozen such tour types, each involving between 2 and 10 cities. For each tour there will be an associated query, and the set of all such queries defines our workload: $\mathcal{Q} = Q_1, Q_2, \ldots$. Notice that for the purpose of our discussion the queries only inform the agents if the desired tour is available or not from a given starting city.

How can one speed up the evaluation of such a workload? We may attempt to precompute all the queries, but their large number makes the total size of their answers too large for us to store.

Obviously the best choice of views depends on the particular statistics available on the database. For illustration purposes, assume the data is such that the graph of connections among the cities is sparse, and that only a small subset of cities are on some cycle. Then, one idea is to precompute a view with the (small) set of cities that belong to some cycle. For example, the view

$$C_5(X_1) :- T(X_1, D_1, X_2), T(X_2, D_2, X_3), T(X_3, D_3, X_4), T(X_4, D_4, X_5), T(X_5, D_5, X_1)$$

computes the set of cities on a cycle of length 5, but ignoring the days of the delivery. The view C_5 can be used to speed up query Q_2, since all answers to Q_2 must be in the view C_5 (it can be easily verified that Q_2 is contained in C_5). But it cannot be used to speed up Q_1, since there is no relationship between cycles of length 2 and those of length 5. For a view to be helpful to the entire workload we need to have access to the views containing all possible cycles, up to some length, say $C_2, C_3, C_4, C_5, \ldots, C_{10}$ and we may not have enough space to materialize all of them independently.

Another idea is to compute a single view with all cities from which we can follow a long chain of cities, say of length 10:

$$V_{10}(X_1) :- T(X_1, D_1, X_2), T(X_2, D_2, X_3), \ldots, T(X_9, D_9, X_{10})$$

Notice that any city that is on a cycle, of any length, is included in V_{10} (i.e., $C_m \subseteq V_{10}$, for every $m \geq 2$) since a chain simply "wraps around" a cycle, but the converse is false. Since V_{10} is a single view, we most probably have enough space to materialize it. More interestingly, all queries in the workload can be sped up by using V_{10}; this can be done in several ways, we only illustrate here one possible rewriting for both Q_1 and Q_2:

$$Q_1(X_1) :- V_{10}(X_1), T(X_1, D, X_2), V_{10}(X_2), T(X_2, D+1, X_1)$$

$$Q_2(X_1) :- V_{10}(X_1), T(X_1, 1, X_2), T(X_2, 1, X_3), T(X_3, 2, X_4), V_{10}(X_4), T(X_4, 2, X_5), T(X_5, 4, X_4), T(X_4, 5, X_1)$$

To see why this results in a speedup, consider the plan $(V_{10} \bowtie T) \bowtie (V_{10} \bowtie T)$ for Q_1: since the result of the join $V_{10} \bowtie T$ is much smaller than the table T, this plan is more efficient than the original plan, $T \bowtie T$.

There is nothing special about the number 10 in V_{10}. We could have used any view V_n defining a chain of length n. In fact, larger values of n will eliminate more false positives, and thus speed up the workload even more.

Our focus in this paper is on the choice of the view(s). First, the example illustrates that it does not suffice to consider only views that are defined as subsets of subgoals of the queries in the workload. In fact, V_{10} is not a subexpression of any query in the workload. Second, the example illustrates that it is not clear where to stop: ever larger values of n seem to produce better and better views V_n. □

As we show later in the paper, it is also unclear how *many* views we need to select. Hence, we are faced with several fundamental questions regarding the view selection problem: (1) which set of views do we need to consider in an optimal view configuration? (2) what is the maximal size of an optimal view configuration? and (3) what is the complexity of the view selection problem?

We show that a key factor affecting the answer to the aforementioned problems is which statistics we may expect to have on the database relations. These statistics are crucial in order to estimate the size of the views we consider to materialize and the cost of evaluating queries over the views. We therefore distinguish two versions of the problem: the Partial Statistics Assumption (PSA) and the Complete Statistics Assumption (CSA). Under PSA, we assume that standard statistics are maintained on the database, and that cost and size estimates are obtained by some estimation function. In contrast, under CSA we assume that we have an oracle that gives us the precise size of any view over the database schema. Note that in practice such an oracle can be based on statistics collected from running queries over the database for some period of time.

After formally defining the view selection problem (Section 2), we begin by considering the problem under CSA (Section 3). We first show that the workload queries provide a double-exponential upper bound on the size of the view definitions we need to consider in an optimal configuration, and as a result, the view selection problem is decidable in quadruple exponential time. But then we show a rather surprising result: in general, an optimal view configuration may include a number of views that is exponential in the size of the query and database schema. As a result, the view selection problem has an exponential-time lower bound. In fact, this result holds even if we further restrict the expressive power of our query language.

Next, we consider the view selection problem under PSA (Section 4). We show here that an optimal solution to the view selection problem always includes a number of views that is bounded by a polynomial in the size of the database schema, the workload of queries, and the binary representations of the relation sizes and the available space bound. We also prove that the size of the view definitions in an optimal configuration is bounded linearly in the size of the query definitions in the workload. The two upper bounds place the view selection problem in the complexity class NP. The results for the PSA assume certain properties of the size estimation function that are general enough to capture size estimators commonly used in practice. While unreasonable choices of size estimation functions break our results, they also make the view selection problem rather uninteresting.

Our results also shed light on the problem of answering queries using views [Hal01]. It is known that given a query Q with n subgoals and a set of views $\mathcal{V}$, there exists an equivalent rewriting of Q using $\mathcal{V}$ only if there exists a rewriting with n subgoals or less. However, it has been an open problem whether an *optimal* rewriting also satisfies the same bound on its size. Our results on the size of an optimal view configuration also establish bounds on the size of an optimal rewriting of a query using a set of views.

1.1 Related work

There has been relatively little theoretical analysis of the view selection problem in the literature to date. In [CG00], Chirkova and Genesereth considered the space requirements for the view selection problem for the context of data warehouse design. They consider certain restrictions under which they show that one can limit the search of an optimal configuration to views that are subexpressions of the queries in the workload. Gupta [Gup97] considers the view selection problem, but he does not model the attributes of the relations being joined. That is, he considers every relation to be a proposition and looks at query plans that are AND-OR graphs over these propositions. As a consequence, his model does not capture selections, projections, or different join predicates. Our work shows that the complexity of the problem crucially depends on modeling these operations. In a later paper [GM99], the authors also consider the cost of view maintenance, but under the same model of inputs. Finally, in both papers, the set of relations in the warehouse is given as part of the input, whereas in our work we're only given the database schema and the workload queries.

In [TS97] Theodoratos and Sellis describe an algorithm for searching a space of candidate warehouses. They do not discuss the complexity of the view selection problem, and their search space does not include warehouses that contain views that are projections on the database relations. As we show in Section 3, considering such views has a significant impact on the complexity of the view selection problem.

Agrawal et al. [ACN00] describe a system for view selection that is incorporated into the Microsoft SQL Server. They present several very effective heuristics for pruning the space of possible view configurations. An important aspect of their work is that they consider the problem of selecting views and indexes simultaneously. Because of that, they do not consider projection views, since those can often (but not always!) be simulated by indexes on other views.

Harinarayan et al. [HRU96] show that the problem of view selection for data cubes is NP-hard, and describe greedy algorithms for approximating an optimal set of views. View-selection for data cubes is further elaborated in [KM99]. In [GHRU97] the work of [HRU96] is extended to include index selection. Other

works that considered algorithms for view selection are [YKL97, BPT97, LH99, ZY99].

2 Problem definition

We consider the view selection problem for the case in which both queries and views may contain joins, projections and equality selections (i.e., conjunctive queries). Furthermore, we consider queries and views under set semantics, rather than bag semantics. Throughout the paper we use the datalog notation for conjunctive queries. Note that in this notation, joins are expressed as multiple occurrences of the same variable. In general, we say that a variable is a *join variable* if it appears in more than one subgoal in the body of the query. Given a database instance D, the *size* of a view V over D is the number of tuples in the answer to V.

Workloads: The appropriate choice of views in a particular context is highly dependent on the set of queries we expect to be given. We model our expected queries by a query workload, which is a set of queries $\mathcal{Q} = Q_1, \ldots, Q_m$, where each query Q_i has an associated non-negative weight, w_i. The weight describes the relative frequency of Q_i within the workload. We require that the weights sum up to 1 ($\sum_{1 \leq i \leq m} w_i = 1$).

View configurations: Given a database schema $\mathcal{R}$ and a workload $\mathcal{Q}$, our goal is to choose a set of views $\mathcal{V}$ to materialize. We refer to a choice of views as a *view configuration* (or *configuration* for short). There are several contexts in which we may be choosing configurations:

1. Performance of query processing: we may choose to materialize a set of views over a database, such that subsequent queries can make use of these views in query processing. Many commercial database systems today support the functionality of answering queries using views.

2. Warehouse design: the goal is to select a set of views to materialize in a data warehouse, on which we expect to process OLAP-style queries. In this case, the query processor of the warehouse must use *only* the selected views in order to answer the queries.

3. Data placement in a distributed setting: we may want to locally cache views on data that is stored in remote locations. When processing queries, the views can be used to reduce the amount of communication between the nodes.

We use the following terminology when referring to view configurations. The *size* of the configuration for a given database instance D is the sum of the sizes of the views evaluated over D. Given a configuration $\mathcal{V}$, we evaluate the cost of answering a workload $\mathcal{Q}$ on $\mathcal{V}$ using a cost model C. Specifically, $C(\mathcal{R}, \mathcal{V}, Q_i)$ is a function that estimates the cost of evaluating the query Q_i given the workload and the sizes of the views in $\mathcal{V}$. The function C uses a set of statistics on the database, which are assumed to be part of the specification of $\mathcal{R}$. We elaborate on these statistics below.

In practice, before we actually materialize the chosen views, we may only be able to *approximate* their sizes. As we see later, the quality of the size estimator function plays a crucial role in the complexity of the view selection problem.

We assume that the function $E(\mathcal{R}, V, \Sigma)$ returns the estimated size of a view V over a database with schema $\mathcal{R}$ and its associated statistics Σ. Note that the function C uses the estimates produced by E, hence the accuracy of the cost depends on the accuracy of both C and E. Given the function C and a configuration $\mathcal{V}$, the cost of the configuration, denoted by $C(\mathcal{R}, \mathcal{V}, \mathcal{Q})$, is the sum $\sum_{Q_i \in \mathcal{Q}} C(\mathcal{R}, \mathcal{V}, Q_i) \times w_i$.

Finally, our goal is to select a view configuration that satisfies a given space constraint. We denote by B an amount of memory allotted for the views, and we assume that B is given via its binary representation. We are now ready to formally define the view selection problem.

Definition 2.1 (**View selection problem**) Let $\mathcal{R}$ be a database schema, B be the available storage space, $\mathcal{Q}$ be a workload on a database described by the schema $\mathcal{R}$, C be a cost estimation function for query processing, and E be a function for estimating the sizes of queries over $\mathcal{R}$. The view selection problem is to find a set of views $\mathcal{V}$ over $\mathcal{R}$ whose total size is at most B and that minimizes $C(\mathcal{R}, \mathcal{V}, \mathcal{Q})$. □ □

A few points are worth noting about the definition before we proceed:

1. The input to the problem includes the database schema and the statistics associated with it. Hence, a solution applies to the *set* of databases obeying these statistics, and not only to a single database instance.

2. The input to the view selection problem does *not* include a database instance. Hence, the complexity results we present later are in terms of the size of the schema and of the workload, not the size of the database.

3. We present several complexity results concerning the view selection problem. While view selection is an optimization problem, the corresponding decision problem that we refer to in the results is: *Is there a view configuration whose cost is less than K?*, for a given number K.

4. It is important to note that materializing all the queries in $\mathcal{Q}$ is not always a possible solution because of the space limitation (even if we ignore the cost of materializing and updating the views). It is, of course, reasonable to assume that B is at least big enough to hold the result of any single query in $\mathcal{Q}$.

2.1 Cost and size estimates

A key difficulty in treating the view selection problem is that we tread a very fine line with the choice of a cost

model. In previous work, the view selection problem left the cost model as a parameter (e.g., [TS97]), or, when actually implemented, relied on the cost estimates of the optimizer (e.g., [ACN00]). Ideally, we would like to leave the cost model as a parameter to the problem, and obtain results that hold for *any* cost model. On the other hand, we show some interesting results when considering certain classes of cost models. This tradeoff affects the following discussion about the inputs to the view selection problem.

Cost model: For the purpose of our discussion, the critical aspect of a cost model is the estimated cost of a join. In our discussion we consider the product cost model for joins. In this cost model, the cost of joining the relations R and S of size M and N respectively is $\alpha \times MN + \beta \times (M + N)$. This cost model faithfully describes most real-world situations:

- For nested-loop joins, the first term is the dominating factor in the cost, as α is a fraction that depends on the number of main-memory pages available.
- For hash or sort-merge joins, the I/O costs are assumed to be proportional to the sum of the sizes of the joined relations, but the main-memory costs are still proportional to the size of the product of the joined relations. Hence, here we assume that α is relatively small.

Finally, we assume that the cost of a selection is that of a scan on the relation, and the cost of a projection on a relation of size N is $N \log(N)$ (though in practice, it is rarely more than $3N$).

When a view selection algorithm examines a particular candidate configuration $\mathcal{V}$, it needs to compute $C(\mathcal{R}, \mathcal{V}, \mathcal{Q})$. In our discussion, we assume the cost can be computed in time polynomial in the size of $\mathcal{Q}$ and the configuration $\mathcal{V}$. In general, query optimization can be exponential in the size of the query, and hence, calculating $C(\mathcal{R}, \mathcal{V}, \mathcal{Q})$ could potentially be exponential in the size of $\mathcal{Q}$. The reasons for our choice is that we want to isolate the effect of the view selection problem from the problem of evaluating a configuration, and that in practice, query optimizers do not perform an exhaustive search of the possible plans.

In estimating costs of configurations, we also need to specify what kinds of query execution plans the optimizer will be considering. We assume standard query plans, where selections are pushed down to the leaves and the projection is the last operation; query plans differ only in the join order. In our discussion we will distinguish between cases in which the optimizer considers only left-linear query plans and cases where it considers all bushy query plans.

Size estimation: A key difficulty in evaluating the cost of a candidate view configuration is that we also need to evaluate the sizes of the views in the configuration, which are, in turn, used as inputs to the cost estimation function C. Hence, a key aspect in the analysis of the view selection problem is specifying which statistics about the database are available as input along with the schema.

The availability of statistics significantly affects the analysis of the view selection problem. Hence, we study two versions of the view selection problem, and we show that they lead to drastically different results. In the first version, discussed in Section 3, we make the *complete statistics assumption* (CSA), in which we assume that we have an oracle that tells us the size of any view over the database. In practice, it may often be possible to obtain such an oracle. In some cases, we can approximate such an oracle by using the results of many runs on the system during which a large body of statistics have been gathered. In other cases, we may be able to evaluate the results of the views and compute their sizes.

In the second version of the problem, we make the *partial statistics assumption* (PSA), where we rely on the size and cost estimators of the query processor, and assume their functions are parameters to the problem. In Section 4 we consider the PSA case under some reasonable assumptions on size estimation functions that are considered in practice.

An important distinction between CSA and PSA is the *monotonicity property* of the size estimator. We say that the size estimation function E is monotone if, whenever a view V_1 is contained in V_2, $V_1 \subseteq V_2$, then $E(\mathcal{R}, V_1, \Sigma) \leq E(\mathcal{R}, V_2, \Sigma)$ for every given statistics Σ. Under CSA size estimators are always monotone, while under PSA they usually are not. This has important consequences on the view selection problem.

3 The complete statistics assumption

In this section we consider the view selection problem under the *complete statistics assumption* (CSA). That is, we assume that we have an oracle that can accurately estimate the size of any view over the database. The main result of the section is that there exist a database schema and a query workload for which the number of views in an optimal configuration is exponential in the size of the schema. As a consequence, we obtain an exponential-time lower bound on the complexity of the view selection problem for CSA.

The first question that we ask about the view selection problem is whether it is even decidable. A priori, it is not even clear which views need to be considered for optimal configurations, and whether there is even a finite number of such views. In fact, in Section 4 we show that in certain cases the number of relevant views could be infinite.

The following theorem, which is an extension of a result in [CG00], shows that view selection is decidable for CSA.

Theorem 3.1 *Let $\mathcal{R}$ be a database schema and $\mathcal{Q}$ be a workload. Under CSA there is an optimal configuration $\mathcal{V}$ for $\mathcal{R}, \mathcal{Q}$ in which each view has a number of subgoals that is at most double-exponential in the total number of subgoals in all queries in the workload $\mathcal{Q}$. As a consequence, the view selection problem is decidable in quadruple exponential time.* □

The proof idea is the following. Given a view V with

a very large number of subgoals, which is used in an optimal configuration, the proof of the theorem constructs another view V' with the following properties: the number of its subgoals is at most double exponential in the size of the schema, it is contained in V ($V' \subseteq V$), and it can replace V in all query rewritings of the queries $\mathcal{Q}$. Here, we use the fact that, under CSA, the size estimation function E is monotone, hence replacing V with V' in all queries will result in a smaller estimated cost. We omit the details of the proof for lack of space, but we illustrate the construction of V' on the Example 1.1. As suggested there, any view defining a chain of length n, V_n, is useful in answering all the queries in the workload, and, apparently, the larger we pick n the better the view configuration is. The new view V' obtained by applying the construction in the proof to V_n, for some large n, returns all cities that are on a cycle of length m, where m is the least common multiplier of all tour lengths in all queries in the workload. (V' is C_m, with the notations in the example.) Notice that picking any smaller value for m would make V' unusable in some queries of the workload. Hence, m is bound by an exponential in $\mathcal{Q}$, namely by the product of the number of variables occurring in every query. More generally, in the proof of Theorem 3.1 the new view V' uses at most as many variables as the product of the number of variables used in the queries $Q_1, Q_2, \ldots$ in the workload: this gives us an exponential bound on the number of variables, resulting in a double-exponential bound in the size of the view definition. The total number of views that have this bound adds another exponential. Finally, an algorithm for finding an optimal solution has to iterate over all subsets of such views, hence quadruple exponential time.

Next, we show that the number of views of an optimal view configuration under CSA can be exponential in the size of the schema and the workload. This is a relatively surprising result, and in fact, none of the algorithms proposed for view-selection would consider such view configurations. Recall that the input to the view selection problem is a database schema with a set of associated statistics. Since we are making the complete statistics assumption, a query processor estimating the cost of query plans has access to accurate estimates of sizes of views over the schema. Hence, in order to show that the number of views of the configuration can be exponential, we need to show a database schema, a workload, and a set of statistics on the database, that will yield such a configuration.

Consider a database schema $\mathcal{R}_1$ that includes two relations R and S of arity n. The workload $\mathcal{Q}_1$ includes three queries:

$q_1(X_1, \ldots, X_n)$:- $R(X_1, \ldots, X_n)$
$q_2(X_1, \ldots, X_n)$:- $S(X_1, \ldots, X_n)$
$q_3(X_1, \ldots, X_n)$:- $R(X_1, \ldots, X_n), S(X_1, \ldots, X_n)$

The first two queries simply ask for the database relations, while the third query asks for their intersection. If we are considering views to be materialized in a database, then the query q_3 suffices for the proof. Queries q_1 and q_2 are needed if we are creating a data warehouse.

The view configuration $\mathcal{V}_1$ includes all the possible projections of $R \cap S$ on a subset of size $n/2$ of its attributes, and one additional view, V_0, which is the projection of $R \cap S$ on any one of its attributes. Note that there are $N = \left(\begin{array}{c} n \\ n/2 \end{array} \right)$ projections of $R \cap S$ on $n/2$ attributes, i.e., the number of views in $\mathcal{V}_1$ is exponential in n. We denote the views by $V_0, \ldots, V_N$.

To establish our result, we show that there exists a set of statistics Σ_1 (whose values depend only on the schema and on the values of α and β in the cost model C) such that an optimal configuration will include *all* the views $V_0, \ldots, V_N$. To do that, we will show that an optimal query plan for the query q_3 will be the one that joins R with V_0, V_1, $\ldots$, V_N successively, and finally joins the result with S. We denote this plan by $\mathcal{P}^*$.

Recall that in the context of CSA, a set of statistics can be viewed as a function that maps every view over $\mathcal{R}_1$ into an integer that specifies its size. We show the existence of Σ_1 as follows. We describe a particular database instance D, and define Σ_1 to be the statistics for D. Hence, Σ_1 represents the set of databases whose statistics agree with those of D. This particular way of constructing the statistics also guarantees that they are consistent, i.e., there exists at least one database instance whose statistics are Σ_1. The details of D are quite involved, and therefore we describe them in a full version of the paper. The database D and the statistics Σ_1 provide the basis for the following theorem.

Theorem 3.2 *The view selection problem under CSA considering only left-linear plans has an exponential-time lower bound.* □

The queries in the workload constructed in the proof of the theorem have a number of head and join variables that depend on the database schema. This is not needed for the results to hold. A variation on the schema $\mathcal{R}_1$ and the statistics $\mathcal{S}_1$ enable us to prove the following result:

Theorem 3.3 *Given a database schema $\mathcal{R}$, a workload $\mathcal{Q}$ and a space constraint B, the view selection problem under CSA and considering only left-linear plans has an exponential lower bound even if all the queries in $\mathcal{Q}$ have a bounded number of variables in their head or if all the queries in $\mathcal{Q}$ have a bounded number of join variables.* □

Remark: Theorems 3.2 and 3.3 hold when we restrict ourselves to left-linear trees. The question of whether the exponential-time lower bound holds when we consider bushy plans remains open. In fact, we can show that an optimal bushy plan for the queries in the workload constructed in the proof of the theorems require only a single view which is a projection of R (or S) on a single attribute.

However, $\mathcal{P}^*$ is still an optimal plan if we consider all bushy plans in which S (or R) is the last relation being

joined (that is, S (or R) is the right child of the root of the join tree). This is an important observation in a distributed query processing context. In this context, we may have R and S stored in different locations, and may store views on S (or on the intersection of R and S) in R's node to perform local filtering before we send data over the network to evaluate the intersection of R and S. Hence, our result implies that the number of local views we may want to store on a remote source may be exponential in the size of the schema. □

Our last result in this section identifies projections as being the culprit for the exponential-time lower bound.

Theorem 3.4 *Given a database schema $\mathcal{R}$, a workload $\mathcal{Q}$, and a space constraint B, if an optimal view configuration includes only projection-free views, then the number of views in the configuration is at most quadratic in the size of the schema and workload.* □

4 The partial statistics assumption

We consider now the view selection problem under the *partial statistics assumptions*, PSA, when only limited statistics over the database are available for estimating the view sizes. In this case our goal is to find a view configuration $\mathcal{V}$ whose total *estimated* size is at most B (the space bound) and that optimizes the *estimated* cost $C(\mathcal{R}, \mathcal{V}, \mathcal{Q})$ of the workload. This scenario is more likely in practice, and is similar to that of an optimizer in a relational database system that has to base its cost estimation on limited database statistics. The main results in this section are that the number of views in an optimal configuration of a view selection problem is bounded by a polynomial, and that the number of subgoals in any view that is part of an optimal solution is bounded linearly in the largest number of subgoals occurring in some query in the workload. The results however hold only under certain assumptions of the selectivity estimation function, which we need to discuss first.

Database systems maintain a collection of statistics on the database to perform size estimates. Statistics range significantly in complexity, from simple numbers to complex histograms. Examples include: the cardinality of a base table; the number of distinct values in a given field; the most frequent values in a given field; an equiwidth histogram on a certain attribute of some base table. Size estimators use heuristics to compute the size of query answers from these statistics. Examples of well-known simple heuristics are those for joins and selections. Given cardinalities N_R and N_S of the tables R and S, the size of $R \bowtie S$ is estimated to be $c \times N_R \times N_S$ where c is the *join selectivity factor*, and the size of a selection $\sigma_b(R)$ is estimated to be $d \times N_R$, where d is the selectivity of the boolean condition b.

The database statistics, denoted Σ, play now a central role in the size estimator function, and we denote by $E(\mathcal{R}, V, \Sigma)$ the estimated size of a view V over the database with schema $\mathcal{R}$ and statistics Σ. Our goal is to analyze the complexity of the view selection problem when the function E is a parameter.

Obviously, artificially chosen functions E could place the view selection problem anywhere between trivial and extremely complex. For example, if E estimates the size of every view with one or more joins to be prohibitively large, then there exists at most one valid view configuration, namely that materializing exactly the base relations (in a warehouse design setting, where we have access only to the views), or not materializing anything (in the query optimization setting, where we have access to the base tables anyway): in this case the problem is trivial. On the other hand E may estimate the size of very complex views, with thousands of columns, as being artificially low, forcing us to inspect an extremely large search space: here the problem becomes overly complex.

Without restricting too much the choice of the estimator function E, we focus our discussion on functions that are more likely to be used in practice. Our results in this section hold even under very generous assumptions about E.

Modeling Statistics: We model database statistics as a collection of two kinds of numbers: factors and cardinalities. Factors are real numbers between 0 and 1, e.g., join selectivity factors, the frequency of a given value in a field, etc. Examples of cardinalities are the number of distinct tuples in a base table, the number of distinct values in a field, the number of values in a certain interval of a histogram, etc. We write $\Sigma = \{c_1, \ldots, c_p, N_1, \ldots, N_q\}$, where $c_1, \ldots, c_p$ are factors and $N_1, \ldots, N_q$ are cardinalities. Statistics are always associated with a particular database schema $\mathcal{R}$. Thus, the numbers p and q in Σ depend on $\mathcal{R}$. The complexity of the view selection problem, which is a function on $\mathcal{R}$, $\mathcal{Q}$ and B, will implicitly also be a function of p and q. Furthermore, statistics can be specific to the particular relations involved (e.g., the join selectivity of R and S), or can be generic and used whenever more specific statistics are not available.

Example 4.1 Consider the schema $\mathcal{R}$ with a binary relation R, a ternary relation S, and the statistics $\Sigma = \{c_{R_2R_1}, c_{S_2}, c_\bowtie, c_\sigma, N_R, N_S\}$. Here $c_{R_2R_1}$ is the join selectivity of $R(X,Y) \bowtie R(Y,Z)$; c_{S_2} is the selectivity factor for a selection on the second attribute of S; $c_\bowtie$ is a generic join selectivity that the function E will use for all other joins; c_σ is a generic selectivity factor for all other selections. N_R, N_S are the cardinalities of R and S respectively. Consider the view:

$$V(X,Y,Z,U) \quad :- \quad R(X,Y), R(Y,Z), \\ S(U, \text{“}Smith'', Z).$$

An estimator may compute the size of the view by the following product:

$$E(\mathcal{R}, V, \Sigma) \quad = \quad c_\bowtie \times c_{R_2R_1} \times c_\sigma \times N_R^2 \times N_S.$$

Thus, the size is a product of the selectivity factors, for the joins and the selections, and of the cardinalities of the base tables. □

Example 4.2 The following example involves a projection. We note that in practice, estimating the size of a

projection (i.e., the `select distinct ...` SQL statement) is much harder than for selections and joins, and there are no widely used robust techniques to address this problem. The following is one of the commonly used estimators. Consider the view:

$$V(X,U) \quad :- \quad R(X,Y), S(Y,Z), T(Z,U)$$

the following size estimation may be used:

$$E(\mathcal{R}, V, \Sigma) \quad = \quad (c_{\bowtie})^2 \times N_{R_1} \times N_S \times N_{T_2}$$

where N_{R_1} is the number of distinct values in the first column of R, N_S is the number of tuples in S, and N_{T_2} is the number of distinct values in the second column of T. □

In summary, our discussion considers the class of *multiplicative size estimators*, which captures estimators used in practice:

Definition 4.1 (**Multiplicative estimators**) A size estimator E on the statistics $\Sigma = \{c_1, \ldots, c_p, N_1, \ldots, N_q\}$ is called *multiplicative* if, for every view V:

$$E(\mathcal{R}, V, \Sigma) = c_1^{\gamma_1} \times \ldots \times c_p^{\gamma_p} \times N_1^{\delta_1} \times \ldots \times N_q^{\delta_q}$$

Here $\gamma_1, \ldots, \gamma_p, \delta_1, \ldots, \delta_q$ are natural numbers that depend on the view V and satisfy the following conditions, where s is the number of subgoals in V: (1) $1 \leq \sum_i \delta_i \leq s$, (2) when V is projection free then $\sum_i \delta_i = s$, and (3) $\sum_i \gamma_i \leq \lambda s$, for some fixed integer $\lambda > 0$. □

Given the results in Section 3 for CSA, one wonders if for PSA we still need an exponential number of views to in an optimal view configuration. The answer is "no", as we shall prove in a moment, but first we notice that PSA introduces a new kind of problem: some multiplicative estimators E may lead to view selection problems that do not have optimal solutions! This is illustrated by the following example. Recall that the cost of a join between two tables of (estimated) sizes M and N is $\alpha MN + \beta(M+N)$.

Example 4.3 Consider the schema $\mathcal{R}$ with two binary relations R and S and the queries:

$$\begin{aligned} Q_1(X,Y) \quad &:- \quad R(X,Y), S(Y,Y) \\ Q_2(X,Y) \quad &:- \quad R(X,Y) \\ Q_3(X,Y) \quad &:- \quad S(X,Y) \end{aligned}$$

Assume the statistics to be $\Sigma = \{c, d, N_R, N_S, N_{S_1}\}$, where N_R, N_S, N_{S_1} are the cardinalities of R, S, and $\pi_1(S)$ respectively, c is the selectivity for $S(Y,Y)$, i.e., $E(\mathcal{R}, S(Y,Y), \Sigma) = c \times N_S$, and d is explained below. Assume a space bound B such that we can store R and S, but not R, S and Q_1, or R, S and $S(Y,Y)$. In particular, $N_R + N_S < B < N_R + N_S + c \times N_S$. Hence, in the warehouse context, we are forced to choose R and S as views to materialize, and have some space left for some additional view(s) to reduce the cost of Q_1. Consider the following infinite sequence of views, for $n = 0, 1, 2, \ldots$:

$$\begin{aligned} V_n(X) \quad :- \quad & S(X, Y_0), S(Y_0, Y_1), S(Y_1, Y_2), \\ & \ldots, S(Y_{n-1}, Y_n) \end{aligned}$$

Suppose E estimates the sizes of the views V_n to be:

$$E(\mathcal{R}, V_n, \Sigma) \quad = \quad d^n \times N_{S_1}$$

Hence d indicates by what factor the cardinality of the semijoin is reduced by each additional S. Since $0 < d < 1$, for all values of n large enough, the estimator will conclude that there is enough space to store V_n in addition to R and S: $N_R + N_S + d^n \times N_{S_1} < B$. Moreover, consider the following rewriting of Q_1:

$$Q_1(X,Y) \quad :- \quad R(X,Y), V_n(Y), S(Y,Y)$$

and the following plan for it: $(R \bowtie V_n) \bowtie S$. As n goes to infinity and the estimated sizes of both V_n and $R \bowtie V_n$ converge to 0, the cost of this plan converges to $\beta(N_R + N_S)$. There is no optimal solution to the problem in this case, since the cost of *any* plan for Q is greater than $\beta(N_R + N_S)$. □

This example shows that Theorem 3.1 from CSA fails under PSA, the reason being that the size estimation function E here is not monotone. Indeed, for every $n \geq 1$, the view $V(Y) :- S(Y,Y)$ is contained in the view V_n ($V \subseteq V_n$). However, for n large enough, the estimates are $E(\mathcal{R}, V, \Sigma) = c \times N_S > d^n \times N_{S_1} = E(\mathcal{R}, V_n, \Sigma)$. In general, as this discussion suggests, it is virtually impossible to find robust estimators that are monotone.

The assumption of large cardinalities: The lack of an optimal solution in the example above is due to the fact that E estimates the cardinalities of certain views V_n to be excessively small (even less than 1, for large n). To avoid such anomalies, we will make the following assumption: *all cardinalities are large.* Technically, this translates into two conditions, one on the function E and the other on the statistics Σ. First, we will assume that the function E always returns a cardinality that is at least L, some lower bound. That is, there exists a multiplicative estimator E' such that:

$$E(\mathcal{R}, V, \Sigma) = \begin{cases} \lceil E'(\mathcal{R}, V, \Sigma) \rceil & \text{when } E'(\mathcal{R}, V, \Sigma) > L \\ 0 \text{ or } L & \text{otherwise} \end{cases}$$

In the gray zone, when the multiplicative formula is less than L, we let the estimator choose whether to believe that the view is empty, and return 0, or that it is non-empty, and return L. For such estimators, we revisit the view selection problem slightly, but allowing only views with a non-zero size estimate to be included in any view configuration[1].

The second technical condition for the large cardinalities assumption is that there exist two numbers

[1] Views with estimated size 0 would, of course, artificially reduce the cost of any plan in which they participate.

c and N such that for every set of statistics $\Sigma = \{c_1, \ldots, c_p, N_1, \ldots, N_q\}$ we have $c \leq \min(c_1, \ldots, c_p)$ and $N \leq \min(N_1, \ldots, N_q, L)$. Recall that the statistics vary with the schema $\mathcal{R}$. Our condition implies that, as the schema increases in complexity, all new cardinalities and/or factors added to Σ are larger than the fixed values N and c. The two theorems below also assume two technical conditions: $c \times N > 1$ and $c^\lambda \times N^\mu > 1$, respectively, for fixed constants λ and μ. These conditions are justified by the same assumption on large cardinalities, and read: "if we apply one selection, or a succession of λ/μ selections respectively, to a table of cardinality N, we still obtain an answer with at least 1 tuple".

A polynomial bound on the number of views: We first prove that, when a solution to the view selection problem exists under PSA, then there exists one with a polynomial number of views. This result should be contrasted to Theorem 3.2, which proves that under CSA certain problem instances require an exponential number of views.

For the remainder of this section, given a view selection problem $\mathcal{R}, \mathcal{Q}, B$, a *view configuration*, $\mathcal{V}$, means one in which every query in the workload $\mathcal{Q}$ is rewritten only in terms of the views, and whose total size does not exceed B.

Theorem 4.1 *Assume $c \times N > 1$. Consider a view selection problem given by $\mathcal{R}$, $\mathcal{Q}$, and B, and assume $\mathcal{V}$ is its optimal view configuration, either under the left-linear plan restriction, or under bushy plans. Then the number of views in $\mathcal{V}$ is bounded by a polynomial in the size of $\mathcal{R}$, $\mathcal{Q}$, and the binary representations of B, c, and N.*

The proof is given in the full version of the paper.

Join-sensitive size estimators: Under our current assumptions for size estimators, some view selection problems may have view configurations but no optimal ones. We show here that, with some additional restrictions on the estimator function, every view selection problem admits an optimal view configuration, if it admits any at all, and that finding such a configuration places the problem in NP.

Referring to the notations in Definition 4.1, we call a multiplicative estimator *join-sensitive* if there exists a positive constant $\mu > 0$ such that

$$\delta_1 + \ldots + \delta_q \geq \mu \times s,$$

where s is the number of subgoals in the view V. Here the constant μ, together with λ, c, and N, are fixed for all database schemas and associated statistics.

The size estimator E in Example 4.3 is not join sensitive, since in the estimated size of V_n there is a single factor that is a cardinality. However, most estimators in practice are join-sensitive since they compute the estimated size bottom up. For example, for V_n, they would first consider the expression $\pi(S \bowtie S \bowtie \ldots \bowtie S)$. The estimator then computes the number of tuples in the join, yielding some expression of the form $f^{n-1} \times N_S^n$, and finally computes the number of tuples in the projection. The result has the form $gf^{n-1} \times N_S^n$, where g and f are some numbers between 0 and 1. Such an estimator is join-sensitive.

The view selection problem always has a solution when the estimator is join-sensitive. Moreover, in searching for a solution, one only needs to consider views whose size is linearly bounded by the largest number of subgoals of any query in $\mathcal{Q}$. This is captured precisely by the following theorem, whose proof is given in the full version of the paper.

Theorem 4.2 *Assume E to be a join-sensitive estimator, and assume $c^\lambda \times N^\mu > 1$. Consider a view selection problem given by $\mathcal{R}$, $\mathcal{Q}$, and B, and assume it admits at least one view configuration. Let s be the largest number of subgoals in any query in $\mathcal{Q}$. Then the following hold, both under the restriction to left-linear join plans and under arbitrary bushy plans: (1) the view-selection problem always has an optimal solution, (2) if a view V is part of an optimal solution then the number of subgoals in V is $O(s)$, and (3) the view selection problem is in NP.*

5 Conclusions

View selection is becoming a critical problem in several data management applications: query optimization, data-warehouse design, data placement in distributed environments, and ubiquitous computing. This paper answered several fundamental questions about the view selection problem: which views need to be considered in an optimal view configuration, what is the cardinality of an optimal view configuration, and what is the complexity of the view selection problem. Our work lays the foundation for both further theoretical analysis and the development of practical algorithms for view selection.

As we have shown, the answer depends critically on whether we can accurately estimate the size of views over the given database. When we have accurate size estimates, i.e., under the complete statistics assumption, we have shown that the cardinality of an optimal view configuration may be exponential in the size of the database schema and query workload. As a result, we have established an exponential-time lower bound on the view selection problem, and a double-exponential upper bound. Under the partial statistics assumption, when we use multiplicative size estimators, we have shown that the cardinality of an optimal view configuration is polynomially bounded, and hence the view selection problem is in NP. We have also shown that under certain conditions, the view selection problem may not have an optimal solution.

Index selection: an important issue in the formulation of the view selection problem is the effect of choosing indexes on the views in the configuration. While our results have not considered the selection of indexes, it is easy to show that the results still hold if we assume that

the number of views in an optimal configuration does not change if we consider indexes.

The intuitive justification for this assumption is that indexes *cannot* increase the number of views needed for an optimal configuration. That is, if an indexed view is part of an optimal plan for a query, then there must be some database instance such that the view is useful in an optimal plan that does not use indexes. The justification for using a view (if it's not needed for correct query semantics) is that using the view reduces the overall cost of the query. This can only happen if the size of the intermediate result is smaller after the join; in fact, it must be sufficiently small that it compensates for the additional cost of performing the join. The presence or absence of an index does not affect the *size* of a join result; hence, the presence or absence of an index is irrelevant as to whether joining with a view reduces overall query execution costs. An index may reduce the cost of performing the join — hence the cost-benefit threshold might change, but the join is only beneficial if it reduces the cardinality in the first place.

Updating costs: another important practical issue is the cost of maintaining the views that have been materialized (though in some contexts it is sufficient to assume views are updated in an off-line process done periodically). In most of the cases we discussed, it suffices to model the cost of updates by assuming some of the queries in $\mathcal{Q}$ are update queries. The subtle issue, however, is that in the presence of updates, the view configuration may contain queries whose sole purpose is to speed up updates, rather than support any of the queries in $\mathcal{Q}$ [RSS96]. In future work, we will extend our analysis to cover such cases.

Acknowledgments

We would like to thank Surajit Chaudhuri, Mike Genesereth, Zack Ives, Henry Kautz and Rachel Pottinger for very stimulating discussions regarding this work.

References

[ACN00] Sanjay Agrawal, Surajit Chaudhuri, and Vivek Narasayya. Automated selection of materialized views and indexes in Microsoft SQL Server. In *Proc. of VLDB*, pages 496–505, Cairo, Egypt, 2000.

[BPT97] Elena Baralis, Stefano Paraboschi, and Ernest Teniente. Materialized views selection in a multidimensional database. In *Proc. of VLDB*, pages 156–165, 1997.

[CG00] Rada Chirkova and Michael Genesereth. Linearly bounded reformulations of conjunctive databases. In *Proc. of DOOD*, pages 987–1001, 2000.

[GHI+01] Steven Gribble, Alon Halevy, Zachary Ives, Maya Rodrig, and Dan Suiu. What can databases do for peer-to-peer? In *ACM SIGMOD WebDB Workshop 2001*, 2001.

[GHRU97] H. Gupta, V. Harinarayan, A. Rajaraman, and J. D. Ullman. Index selection for OLAP. In *Proc. of ICDE*, pages 208–219, 1997.

[GM99] H. Gupta and I. S. Mumick. Selection of views to materialize under a maintenance cost constraint. In *Proc. of ICDT*, pages 453–470, 1999.

[Gup97] H. Gupta. Selection of views to materialize in a data warehouse. In *Proc. of ICDT*, pages 98–112, 1997.

[Hal01] Alon Y. Halevy. Answering queries using views: A survey. *To appear in the VLDB Journal*, 2001.

[HRU96] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proc. of SIGMOD*, pages 205–216, 1996.

[ILM+00] Z. Ives, A. Levy, J. Madhavan, R. Pottinger, S. Saroiu, I. Tatarinov, E. Jaslikowska, and T. Yeung. Self-organizing data sharing communities with SAGRES: System demonstration. In *Proc. of SIGMOD*, page 582, 2000.

[KM99] Howard J. Karloff and Milena Mihail. On the complexity of the view-selection problem. In *Proc. of PODS*, pages 167–173, Philadelphia, Pennsylvania, 1999.

[Kos00] Donald Kossmann. The state of the art in distributed query processing, 2000. Submitted for publication.

[LH99] Minsoo Lee and Joachim Hammer. Speeding up warehouse physical design using a randomized algorithm. In *Proc. Int'l Workshop on Design and Management of Data Warehouses (DMDW-99)*, 1999.

[RSS96] K.A. Ross, D. Srivastava, and S. Sudarshan. Materialized view maintenance and integrity constraint checking: trading space for time. In *Proc. of SIGMOD*, pages 447–458, 1996.

[TS97] Dimitri Theodoratos and Timos Sellis. Data warehouse configuration. In *Proc. of VLDB*, pages 126–135, Athens, Greece, 1997.

[YKL97] J. Yang, K. Karlapalem, and Q. Li. Algorithms for materialized view design in data warehousing environment. In *Proc. of VLDB*, pages 136–145, Athens, Greece, 1997.

[ZY99] Chuan Zhang and Jian Yang. Genetic algorithm for materialized view selection in data warehouse environments. In *Proc. Int'l Conf. on Data Warehousing and Knowledge Discovery (DaWak-99)*, 1999.