Designing Views to Optimize Real Queries
(extended abstract)

Rada Chirkova*
Computer Science Department
North Carolina State University, NC 27695

Abstract

This paper considers the following problem: given a query workload, a database, and a set
of constraints, design a set of views that give equivalent rewritings of the workload queries and
globally minimize the evaluation costs of the workload on the database under the constraints. We
refer to this problem as “view design for query performance,” or “view design” for short; sets of
views that satisfy the requirements are called “optimal viewsets.” It has been shown that for the
storage-limit constraint and under set semantics, the view-design problem is decidable and has an
exponential-time lower bound for conjunctive queries, views, and rewritings.

In this paper we explore view design for workloads of conjunctive queries, under a range of
assumptions about databases and query computation. We first look at conjunctive views and
rewritings under bag and bag-set semantics. We show that under bag semantics, the view-design
problem is in NP. Surprisingly, under bag-set semantics we cannot use past work on bag-set query
equivalence [8] to determine whether a rewriting of a query in terms of views is equivalent to the
query. In addition, the view-design problem under bag-set semantics is characterized by the same
complexity and size of the search space of optimal views as its counterpart under set semantics.
We then explore special cases and more expressive view and rewriting languages. We show that for
workloads of queries without self-joins, the search space of optimal views is dramatically smaller
than that for queries with self-joins, under both set and bag-set semantics. Further, we show that
the view-design problem, for queries with or without self-joins, is decidable (1) for disjunctive views
and rewritings, and (2) for views with semipositive negation and without self-joins.

1 Introduction

In this paper we consider the problem of improving the performance of query workloads on relational
databases, by using the well-known methodology of answering queries using views [18, 19]. To improve
query performance, it is common to precompute and store, or materialize, views. Computing queries
efficiently using materialized views is important in many data-management applications, notably in
the areas of query optimization, data warehousing, and information integration [2, 3, 6, 24, 25, 26]. To
truly optimize query performance, it is critical to design and materialize the “right” views. The paper
focuses on the following problem: given a query workload, a database, and a set of constraints, design a
set of views that give equivalent rewritings of the workload queries and globally minimize the evaluation
costs of the workload on the database under the constraints. We refer to this problem as “view design
for query performance,” or view design for short; sets of views that satisfy the requirements are called
optimal viewsets.

The results we report in this paper build on past work in designing views to materialize. The
problem of automating the process of finding useful views to materialize has received a lot of attention
in the context of efficient query evaluation, and has been studied extensively under the name of view
selection [3, 4, 15, 16, 17, 20, 24, 25, 27]. View selection may involve choosing a set of indexes on the

*Address: North Carolina State University, Computer Science Department, Campus Box 7535, Raleigh, NC 27695-7535, USA. Phone: (919) 513-3506. Fax: (919) 513-4357. Email address: chirkova@esc.ncsu.edu
views [3, 15, 16] — some papers, e.g., [7], treat index selection separately — and may consider the
cost of updates to the views [15, 17, 22, 24, 25, 27].

It is important to realize that research in view selection has been conducted under the tacit
assumption that it is only necessary to consider for materialization the views that are given or described
in the problem input. However, in general, to truly optimize query performance, it may be necessary
to come up with new view definitions, in other words, to invent new views. Some theoretical work has
been done on characterizing globally optimal views for queries under a storage limit, which is an upper
bound on the disk space available for storing the materialized views [9, 10, 11, 12]. In particular, it
has been shown that for the storage-limit constraint and under set semantics, the view-design problem
is decidable and has an exponential-time lower bound for conjunctive queries, views, and rewritings.

In this paper we explore view design for workloads of conjunctive queries, under a range of assump-
tions about databases and query computation. The ultimate goal of this research is to devise efficient
and scalable algorithms to design views that would ensure optimal or near-optimal performance of
workloads of frequent and important queries, on database instances that change over time. (Because
optimal query performance is unlikely to be achieved in practice for many or most problem setups,
the focus, for those problem setups, is on near-optimal performance.) The results we report in this
paper bring us a step closer to this goal, by giving us an understanding of the structure of the space
of optimal views for certain subproblems of the view-design problem.

The contributions of the paper are as follows. We explore view design for workloads of conjunctive
queries. After giving, in Section 2, a formal specification of the view-design problem for conjunctive
workloads, in Section 3 we explore view design with conjunctive views and rewritings, under bag and
bag-set semantics. We show that under bag semantics, the view-design problem is in NP. Surprisingly,
under bag-set semantics we cannot use past work on bag-set query equivalence [8] to determine whether
a rewriting of a query in terms of views is equivalent to the query. In addition, the view-design
problem under bag-set semantics is characterized by the same complexity and size of the search space
of optimal views as its counterpart under set semantics. Next, in Section 4, for the special case of
workloads of conjunctive queries without self-joins, we show that the search space of optimal views is
dramatically smaller than the search space of optimal views for queries with self-joins, under both set
and bag-set semantics. In Section 5 we examine more expressive languages for views and rewritings, in
designing views for conjunctive query workloads with or without self-joins. We show that the problem
is decidable (1) for disjunctive views and rewritings, and (2) for views with semipositive negation and
without self-joins of positive subgoals.

2 Preliminaries

2.1 Problem specification

In this section we give a formal specification of the view-design problem. We use the book [1] for
notation and standard definitions. Given a finite set, or workload, \( Q \) of queries defined on the schema
\( D \) of a database \( D \), and given a set \( C \) of constraints, we want to define, on the schema \( D \), views
\( V = \{V_1, \ldots, V_l\} \), to be precomputed and stored, or materialized, on the database \( D \). We design the
views offline, that is, before any queries in \( Q \) are executed. A set \( V \) of views is a solution for a given
problem input \( I = (Q, D, C) \) if (1) the views in \( V \) give equivalent rewritings of all queries in \( Q \) (i.e.,
the answer to each query is preserved on all databases [18, 19]), and if (2) the relations for the views
\( V \), materialized on the database \( D \), satisfy the constraints \( C \) and reduce the evaluation costs of the
workload \( Q \) on \( D \). A solution \( V_1 \) for a problem input \( I \) is at least as good as another solution \( V_2 \) for \( I \)
if the solution \( V_1 \) reduces the costs of computing the input query workload \( Q \) on the input database \( D \)
more than does the solution \( V_2 \). A solution \( V \) for a problem input \( I = (Q, D, C) \) is an optimal solution
for \( I \) if \( V \) globally minimizes the costs of computing the workload \( Q \) on the database \( D \) under the
constraints \( C \). The view-design problem is to find an optimal solution for a given problem input.
We assume that in a query workload $Q$, each query $Q_i$ has an associated weight $w_i$, which gives the relative frequency of the query in the workload. The cost $c(Q_i)$ of a query $Q_i$ on a database $D$ is determined using standard left-linear query plans: selections are pushed down to the leaves of the plans, projection is the last operation, and the right input of any join is a stored relation. We use the sum-cost model for joins, where the cost of joining two relations is proportional to the sum of the sizes of the relations and of the size of the join result — as in, for example, hash join [14]. The cost of computing a rewriting of a query in terms of views is determined using estimates for the sizes of the views and intermediate relations in the chosen query plan on the database $D$. The cost of the workload $Q$ on the database $D$ is the weighted sum $\sum_{i=1}^{n} w_i \times c(Q_i)$. We consider query computation under set, bag, and bag-set semantics [8], and assume that indexes are not used in the computation.

In this paper we consider the view-design problem for workloads of safe conjunctive queries [1] and a single storage-limit constraint, which is the amount of disk space available for storing materialized views. (But see Section 3 for a more general case.) We consider views defined by safe conjunctive queries, disjunctive queries, or queries with semipositive negation. A disjunctive query is a union of two or more conjunctive queries; a query with semipositive negation — in semipositive datalog — is defined by a conjunction of positive and negative literals for the relations in the database schema $D$. We consider query rewritings that are either conjunctions of views (conjunctive rewritings) or unions of conjunctions of views (disjunctive rewritings).

There is an obvious omission in our formulation of the view-design problem: To find a globally optimal viewset for a problem input, it is necessary to have exact estimates of the sizes of the views considered for materialization, as well as of the costs of answering the input queries using the views. It has been shown [12] that the view-design problem is in NP for conjunctive queries, views, and rewritings under the storage-limit constraint, provided standard query-optimizer functions, under certain additional assumptions, are used to estimate the sizes of views and intermediate results in query computation. However, in that approach, there is no guarantee that the estimates of the sizes of relations are exact or even acceptable; it can be shown that, for certain families of views and rewritings, the view-size estimates used in [12] can be arbitrarily far from the exact sizes of the views on certain databases. In this paper, to study the complexity of view design, we separate the view-design problem from the problem of obtaining reliable view-size estimates, by assuming that for any problem input with a database $D$, we can instantaneously obtain an exact size estimate for any relation defined on $D$. For this reason, in the complexity results we discuss here, we do not take into consideration the size of the input database $D$. At the same time, to make possible any practical application of designing globally optimal viewsets for query performance, it is imperative to come up with efficient and scalable algorithms that would give reliable estimates of the sizes of views defined on a database, for common and important view-definition languages.

### 2.2 Two types of views in query rewritings

In this section, we use some definitions and discussion from [13]. There are two types of views that can be used to rewrite a conjunctive query: containment-target views and filtering views. The two types of views can be distinguished, under set semantics, by examining containment mappings from the query to the expansion of the rewriting. Consider an example.

**Example 2.1** Let $Q$ be a query that is a natural join of two binary relations, $R(A, B)$ and $S(B, C)$.

It is possible to answer the query $Q$ using its rewriting $Q'$ in terms of views $V_1$, $V_2$, and $V_3$:

$$
Q : \quad q(X, Y, Z) \quad \Rightarrow \quad r(Y, X), s(X, Z). \\
Q' : \quad q'(X, Y, Z) \quad \Rightarrow \quad v_1(X), v_2(Y, X), v_3(X, Z). \\
Q'' : \quad q''(X, Y, Z) \quad \Rightarrow \quad r(T, X), s(X, W), r(Y, X), s(X, Z). \\
$$

Here, $Q''$ is the expansion of the rewriting $Q'$; we can use $Q''$ to show that the query $Q$ and its rewriting $Q'$ are equivalent under set semantics [5].

\[ \square \]
Table 1: Stored relations $R$ and $S$ in Example 2.1

<table>
<thead>
<tr>
<th></th>
<th>$A$</th>
<th>$B$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>$B$</th>
<th>$C$</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>9</td>
<td></td>
</tr>
</tbody>
</table>

Intuitively, in the computation of a query using a rewriting, each containment-target view provides, or “covers,” at least one query subgoal. Covering all query subgoals is enough to produce an equivalent rewriting of the query. Formally, a view $V$ is a containment-target view for a query $Q$ if there exists a rewriting $Q'$ of $Q$ ($Q'$ uses $V$), and there is a containment mapping from $Q$ to the expansion $Q''$ of $Q'$, such that the view $V$ provides the image of at least one subgoal of $Q$ under the mapping. In Example 2.1, the view $V_2$ covers the subgoal $r(Y, X)$ of the query $Q$, whereas the view $V_3$ covers the query subgoal $s(X, Z)$. Thus views $V_2$ and $V_3$ are containment-target views for the query $Q$.

A view is a filtering view for a query if it is not a containment-target view. Intuitively, filtering views are not necessary in constructing query rewritings, in the sense that those views do not cover query subgoals. However, under set semantics, there could exist some query plan, in which a filtering view removes — filters out — dangling tuples from at least one join input in the plan, which may reduce the evaluation time of the query. In Example 2.1, using the filtering view $V_1$ in the rewriting $Q'$ removes from the relation $V_2$ all tuples that cannot join with tuples in $V_3$. For instance, if the relations $R$ and $S$ are as shown in Table 1, the relation $V_1 = \{(2)\}$ will be able to filter the tuple $(4, 5)$ from the relation $V_2$, which is the same as $R$. If the number of such dangling tuples in $V_2$ is large, using the filtering view $V_1$ in the rewriting $Q'$ can reduce the evaluation time of the query $Q$.

We show in the paper the different roles of the two types of views under bag and bag-set semantics.

3 Designing Views under Bag and Bag-Set Semantics

It has been shown that for the storage-limit constraint and under set semantics, the view-design problem is decidable and has an exponential-time lower bound for conjunctive queries, views, and rewritings, and that the size of the search space of optimal views is at most doubly-exponential in the size of the input query workload [9, 12]. In this section we consider view design under more realistic assumptions about databases and query computation, by looking at bag and bag-set semantics. We still restrict ourselves to purely conjunctive problem setups, that is, in this section we consider only conjunctive queries, views, and rewritings. We show that under bag semantics, the view-design problem is in NP. Surprisingly, under bag-set semantics we cannot use past work on bag-set query equivalence [8] to determine whether a rewriting of a query in terms of views is equivalent to the query. We show that under bag-set semantics, the view-design problem is characterized by the same complexity and size of the search space of optimal views as its counterpart under set semantics.

3.1 The bag-semantics case

We first show that under bag semantics, a query is equivalent to its rewriting using views if and only if the expansion of the rewriting is formally equivalent to the query. (Under bag semantics, two conjunctive queries are “formally equivalent” iff they are isomorphic [8].) This result looks trivial. It is, indeed, unclear why we should put any effort into proving this result at all — until we move on to exploring answering queries using views under bag-set semantics. It turns out that under bag-set semantics, the analog of this result does not hold: For some queries and their equivalent rewritings in terms of views, the expansion of the rewriting is not equivalent to the query under bag-set semantics.

We first show the following. Suppose we want to compute the answer to a query on a database under bag semantics, using a rewriting of the query in terms of views. One way to compute the answer is to first compute the relations for the views on the database and then to use these view relations in
the definition of the rewriting. An alternative plan is to use the expansion of the rewriting and the original stored relations in the database. Proposition 3.1 below says that under bag semantics and on any database instance, both methods of computing the rewriting result in the same bag of tuples.

**Example 3.1** Consider again the query $Q$, its rewriting $Q'$ in terms of views $V_1$ through $V_3$, and the expansion $Q''$ of $Q'$, all given in Example 2.1; consider the database in Table 1. To obtain the answer $\text{ans}(Q')$ to the rewriting $Q'$, we can compute the relations for the three views and then take their join, or we can compute the expansion $Q''$ of the rewriting on the stored relations $R$ and $S$. On the database shown in Table 1, these two ways of computing the relation for the rewriting $Q'$ result, under bag semantics, in the same relation, which comprises sixteen tuples. We will see in Section 3.2 that on this database, the answers to $Q'$ and $Q''$ are not the same under bag-set semantics.

**Proposition 3.1** The following holds for any conjunctive query and for any conjunctive rewriting of the query in terms of conjunctive views. Consider two plans for computing the rewriting of the query on a database. The first plan is to compute all view relations and then to use these view relations in the definition of the rewriting using the views. The second plan is to compute the answer using the original stored relations and the definition of the expansion of the rewriting. Under bag semantics, the outcomes of the two plans on any database are the same bag of tuples.

From Proposition 3.1 and from [8] (two conjunctive queries are equivalent under bag semantics iff they are isomorphic), we obtain the result that was mentioned in the beginning of this section: In the purely conjunctive problem setup, a query is equivalent to its rewriting in terms of views iff the expansion of the rewriting is isomorphic to the query. (From this result it follows that the query $Q$ in Example 2.1 is not equivalent to its rewriting $Q'$ under bag semantics.) We use this result to obtain the following characteristic of equivalent rewritings of queries using views under bag semantics:

**Corollary 3.1** Under bag semantics, for any conjunctive query and for any equivalent conjunctive rewriting of the query in terms of conjunctive views, the bodies of the definitions of the views, taken together, constitute a partition of the body of the query.

From this corollary it follows that under bag semantics, for any query and for any equivalent rewriting of the query using views, each view in the rewriting is a containment-target view for the query — in other words, filtering views cannot be used in equivalent rewritings of queries under bag semantics. (This observation helps us see why, in Example 2.1, the query $Q$ is not equivalent to its rewriting $Q'$ under bag semantics. Removing the filtering view $V_1$ from the rewriting $Q'$ would result in a rewriting that is equivalent to the query $Q$ under bag semantics.) In addition, from Corollary 3.1 it follows that under bag semantics, the definitions of all possible views in all equivalent rewritings of a query can be obtained by partitioning the query subgoals. We conclude that:

**Theorem 3.1** Under bag semantics and for any purely conjunctive problem setup with any type of constraints: (1) The view-design problem is decidable, (2) the size of the search space of optimal views is singly-exponential in the size of the input query workload, and (3) the problem is in NP.

It follows that for any purely conjunctive problem setup with any type of constraints — including the storage-limit constraint and the view-maintenance-costs constraint (an upper bound on the costs of updates to the view relations) — we can always find a globally optimal viewset, by considering, for each input query separately, all sets of views whose bodies partition the subgoals of the query. In each equivalent rewriting, (1) the total number of views, and (2) the number of subgoals in each view, are each bounded from above by the number of subgoals of the query (cf. Lemma 3.5 in [21]).

### 3.2 The bag-set-semantics case

We now study view design under bag-set semantics. Using the fact that query equivalence under bag-set semantics entails equivalence under set semantics [8], we obtain the following two results.
One, the view-design problem is decidable for the purely conjunctive problem setup under bag-set semantics, and two, the size of the search space of optimal views, under bag-set semantics, is at most doubly-exponential in the size of the input query workload. In this section we show that under bag-set semantics, the view-design problem is characterized by the same complexity and size of the search space of optimal views as its counterpart under set semantics.

We first show that under bag-set semantics we cannot use past work on bag-set query equivalence [8] to determine whether a rewriting of a query in terms of views is equivalent to the query.

**Example 3.2** Consider again the database schema, query definition, and the rewriting of the query in terms of views given in Example 2.1, and the database $D$ given in Table 1. Under bag-set semantics, materialized views are sets. For this reason, the relation for the view $V_1$ on $D$ is a singleton set: $V_1(D) = \{(2)\}$. Using this relation, we obtain that under bag-set semantics, the answer to the rewriting $Q'$ on the database $D$ is a bag of four tuples. (The answers to $Q'$ and $Q$ are the same on the database $D$; we will see later that $Q$ is equivalent to $Q'$ under bag-set semantics.)

At the same time, the answer to the expansion $Q''$ of the rewriting $Q'$ on the database $D$ is a bag of sixteen tuples. We conclude that the answers to $Q'$ and $Q''$, computed under bag-set semantics, are not the same on the database $D$.

From Example 3.2 we obtain the following result:

**Proposition 3.2** There exist conjunctive queries and their conjunctive rewritings in terms of conjunctive views, such that under bag-set semantics, the rewriting is not equivalent to its expansion.

From this result it follows that past work on query equivalence under bag-set semantics [8] cannot be used to obtain equivalent rewritings of queries using views. The reason is simple: Materialized views are sets, and they are used as sets when a query is computed using the definition of its rewriting. At the same time, in the expansion of a query rewriting, the expansions of individual views may turn out to be nontrivial bags under bag-set semantics. It follows that under bag-set semantics, even if the answers to a rewriting and to its expansion are the same as sets on some database, they are not necessarily the same as bags on that database.

It is easy to see that the duality in the behavior of views — views are sets when used in rewritings, but are not necessarily sets when used in the expansions of the rewritings — can happen only when the relation for a view, computed under bag-set semantics, can be a nontrivial bag. (For example, the relation for the view $V_1$ in Example 2.1, computed under bag-set semantics, comprises four tuples on the database in Table 1.) At the same time, the process of materializing the relation for any view involves removing all duplicates from the relation. We further observe that when a conjunctive relation is computed under bag-set semantics, the relation can be a nontrivial bag only when the projection operation is used in the computation. In other words, if there are no projections in the definition of a relation, then the relation is a set on any database. In particular, when the definition of a conjunctive query does not have projections, then the answer to the query, under bag-set semantics, is always a set. We use this observation to obtain the following result:

**Proposition 3.3** Assuming conjunctive queries, views, and rewritings, consider a query and its rewriting in terms of views, such that all variables of the query and of the rewriting are distinguished. If the query and the rewriting are equivalent under set semantics, then they are also equivalent under bag-set semantics.

(A variable in a query is distinguished if it appears in the query’s head.) Notice that when view definitions do not have projections but query definitions do, this result is no longer true; it is easy to construct a counterexample to Proposition 3.3 by using Example 7.10 in [8]. Also notice that under bag-set semantics, it is not necessarily true that expansions of equivalent rewritings are equivalent to the queries. The reason is, expansions of query rewritings are computed using expansions of view definitions, which in that computation are not necessarily sets under bag-set semantics. For instance,
from Proposition 3.3, the query $Q$ and the rewriting $Q'$ in Example 2.1 are equivalent under bag-set semantics. At the same time, the expansion $Q''$ of the rewriting $Q'$ is not equivalent to the query $Q$ under bag-set semantics, as can be seen in Example 3.2.

Even though Proposition 3.3 covers only an extremely limited special case of equivalence under bag-set semantics, it allows us to obtain the following result:

**Theorem 3.2** The view-design problem with the storage-limit constraint has an exponential-time lower bound under bag-set semantics, for conjunctive queries, views and rewritings.

This result draws on an example that was devised for view design with the storage-limit constraint under set semantics [9]. In that example, neither the queries nor their equivalent rewritings have nondistinguished variables, and there exist databases on which the number of views in an optimal viewset is exponential in the size of the query workload.

We can single out two more special cases under bag-set semantics. First, when all variables in both queries and views are distinguished, we obtain that (1) filtering views cannot be used in rewritings that are equivalent to the queries under bag-set semantics, and (2) the view-design problem is in NP, under bag-set semantics and for all types of constraints in problem inputs. This case still deals exclusively with queries whose answers, under bag-set semantics, are sets on all databases. Interestingly, we can also say something about queries that do have nondistinguished variables — and whose answers, for that reason, may be nontrivial bags on some databases under bag-set semantics. In contrast with the bag-semantics case, we can use filtering views to obtain equivalent rewritings of such queries under bag-set semantics. Intuitively, because all materialized views are sets, adding a filtering view to an equivalent rewriting of a query cannot create undesirable duplication of tuples in the computation of the rewriting under bag-set semantics. For this reason, using Example 1.1 from [12], we obtain the following. Under bag-set semantics, the view-design problem is characterized by the same upper bound on the size of the search space of optimal views — doubly-exponential in the size of the input query workload — as its counterpart under set semantics.

### 4 Queries Without Self-Joins

We have seen that under both set and bag-set semantics, in view design for workloads of conjunctive queries under the storage-limit constraint, the size of the search space of optimal conjunctive views is at most doubly-exponential in the size of the input query workload. In this section, we tighten this bound, for the special case of queries without self-joins. (We still consider conjunctive queries, views, and rewritings only.) We show that for any problem input where queries do not have self-joins, under set or bag-set semantics, we can construct a search space of views that includes all views in at least one optimal viewset and is, at the same time, relatively small. More precisely, the size of this space of views is at most singly-exponential in the size of the query workload, a nice contrast to the doubly-exponential size in the general case of queries with self-joins. (At the same time, using an example from [9], it can be shown that for queries without self-joins and for the storage-limit constraint, the view-design problem still has an exponential-time lower bound, under both set and bag-set semantics.)

We first show that, if queries in a problem input do not have self-joins, then at least one optimal solution to the view-design problem, under set semantics, has views with relatively short definitions and uses each view at most once in the rewriting of each individual workload query. (The existence of such optimal solutions cannot be guaranteed for queries with self-joins.)

**Proposition 4.1** Under set semantics, for conjunctive queries without self-joins and for conjunctive views and rewritings, if the view-design problem under the storage-limit constraint has a solution, then it has at least one optimal solution where each view is defined as a subexpression of one of the input queries and, in the rewriting of each individual query, each view in the viewset is used at most once.

Here is an intuition for the proof of the proposition. Consider any viewset that is an optimal solution for some problem input where queries do not have self-joins. We can change the viewset by
replacing each view \( V \) that has self-joins, with a view \( V' \) that does not have self-joins, as follows. Each self-join in a view \( V \) can be eliminated by applying to the view definition the most general unifier [1] of the subgoals in the self-join. Doing this operation for all self-joins in the view \( V \) results in a view \( V' \) that is contained in \( V \) but can, at the same time, replace each occurrence of \( V \) in the equivalent rewriting of any query in the problem input. It follows that replacing \( V \) with \( V' \) can only result in a better solution for the same problem input. In addition, once a view \( V' \) (either containment-target or filtering) has been used in a query rewriting, reusing the view in the rewriting leads to either higher costs in the computation of the query, or to a rewriting that is not equivalent to the query.

As query containment under bag-set semantics entails query containment under set semantics, from Proposition 4.1 we obtain the following result:

**Theorem 4.1** Under set or bag-set semantics, for conjunctive queries without self-joins and for conjunctive views and rewritings, the following holds. If the view-design problem under the storage-limit constraint has a solution, then we can construct a search space of views that contains at least one optimal solution for the problem; the size of this search space of views is at most singly-exponential in the size of the input query workload.

In Section 3.1 we saw that the view-design problem is in NP under bag semantics, for queries with or without self-joins. Under set and bag-set semantics, the problem has an exponential-time lower bound even when queries do not have self-joins, because for some problem inputs, the number of views in an optimal viewset can be exponential in the size of the input query workload [9, 12].

## 5 More Expressive Views and Rewritings

In our search for optimal solutions to the view-design problem, for workloads of conjunctive queries and under the storage-limit constraint, we have so far restricted ourselves to conjunctive views and rewritings. We have seen that although this version of the problem is decidable, it has unacceptable complexity under set and bag-set semantics. In this section we widen our scope, by obtaining the first results on view design for query performance with more expressive languages for views and rewritings. The main results of this section are as follows: The view-design problem, for workloads of conjunctive queries with or without self-joins, is decidable (1) for disjunctive views and rewritings, and (2) for views with semipositive negation and without self-joins of positive subgoals.

### 5.1 Disjunctive views and rewritings

In this section, we use our results [13] for the problem of designing views to rewrite conjunctive queries, with the objective of minimizing the total size of the materialized views. We use the observations reported in [13] to obtain the following results for view design for query performance, with the storage-limit constraint and under set semantics:

**Observation 5.1** Suppose we can use conjunctions of disjunctive views in query rewritings. Then there exist conjunctive queries, storage limits, and databases, such that the most efficient equivalent rewriting of the query on the database under the storage limit uses nontrivial disjunctive views.

Intuitively, for some queries and databases, one can choose the value of the storage limit to ensure the following two conditions. One, for any conjunctive viewset that gives an equivalent rewriting of the query, the viewset does not satisfy the storage limit. Two, there exist nontrivial disjunctive views that satisfy the storage limit and give an equivalent rewriting of the query. Interestingly, when we disallow queries with self-joins in problem inputs, in the search space of disjunctive views there always exists at least one optimal viewset that consists of conjunctive views only:

**Proposition 5.1** For the view-design problem for workloads of conjunctive queries without self-joins, under the storage-limit constraint and assuming conjunctive rewritings only, there exists at least one optimal solution in the space of disjunctive views, such that all views in the solution are conjunctive.
Using results in [13] and [23], for the view-design problem with disjunctive views and rewritings we obtain that the problem is decidable and that conjunctive rewritings are enough:

**Theorem 5.1** Under set, bag, and bag-set semantics, the following holds for the view-design problem with workloads of conjunctive queries and the storage-limit constraint, assuming disjunctive views and rewritings. (1) The problem is decidable, and (2) for any problem input \( \mathcal{I} \), there exists an optimal viewset \( \mathcal{V} \), such that equivalent rewritings of all queries in \( \mathcal{I} \) are conjunctions of views in \( \mathcal{V} \).

Intuitively, under set semantics and given a query workload, it is possible to construct a finite number of disjunctive viewsets that give all equivalent rewritings of the workload queries, provided the total size of views in each viewset does not exceed the size of any optimal conjunctive viewset for the workload. We extend this result to the cases of bag and bag-set semantics by recalling that query equivalence in each case entails query equivalence under set semantics [8]. In addition, for set, bag, and bag-set semantics we observe that for any problem input \( \mathcal{I} \) with conjunctive queries it is possible to construct a search space of disjunctive views that includes at least one optimal solution for \( \mathcal{I} \). The number of views in that search space is at most triply-exponential in the size of the query workload in \( \mathcal{I} \). Notice that for view design under the view-maintenance-costs constraint, nontrivial disjunctive views probably give better solutions than conjunctive views, because a disjunctive view requires at most the same number of updates as its conjunctive components.

### 5.2 Views with negation

In view design under a storage limit, there exist problem inputs, such that the number of subgoals in optimal views may be exponential in the size of the input queries, or the number of views in optimal rewritings may be exponential in the size of the input query workload [12]. For these problem inputs we have designed sets of views *with negation*, with the following properties. (Due to the lack of space, we cannot show the examples in this abstract.) One, the views with negation give better solutions to the above inputs than any conjunctive views. Two, for each view in these solutions with negation, the number of subgoals in the view definition is polynomial in the number of subgoals of the queries in the problem inputs. Finally, the number of views in each solution with negation is polynomial in the size of the query workload. These properties are, of course, not surprising: When we allow more expressive views in query rewritings, we should expect that each such view would be able to “do more and with less effort” in the rewriting. At the same time, because it is possible that views with negation will let us do away with the exponential-time lower bounds in view design, studying views with negation should become an important research direction in practical view design for query performance.

We now report two preliminary results in view design with conjunctive queries under a storage limit and with views with semipositive negation [1]. These results hold under set semantics.

**Proposition 5.2** For any conjunctive query \( Q \) that is not trivially empty, if a view with semipositive negation is used in a conjunctive rewriting of \( Q \), then the rewriting is not equivalent to the query.

Intuitively, it is always possible to construct a database instance on which a conjunction of views gives an empty answer whenever at least one of the views is defined using semipositive negation. It follows that to construct equivalent rewritings of conjunctive queries in terms of views with negation, one has to look at rewritings that are more expressive than conjunctive rewritings. Our examples with views with semipositive negation [11], as well as the examples mentioned in the beginning of this section, use disjunctive rewritings.

**Proposition 5.3** The view-design problem is decidable for conjunctive queries, storage-limit constraint, and safe views that are defined using semipositive negation and do not have self-joins of positive subgoals.

Intuitively, it is possible to finitely enumerate all safe views with semipositive negation that can be defined on a given database schema and do not have self-joins of positive subgoals. In our future
work on designing views with negation for query performance, we plan to use past work on expressions with negation ([11] and [23] in particular) to explore the decidability and complexity of the problem for views with more general types of negation.

References