Designing Views to Efficiently Answer *Real* SQL Queries

Foto Afrati[1], Rada Chirkova[2], Manolis Gergatsoulis[3], and Vassia Pavlaki[1]

[1] Department of Electrical and Computing Engineering,
National Technical University of Athens (NTUA), 15773 Athens, Greece ***
{afrati,vpavlaki}@softlab.ntua.gr
[2] Computer Science Department, North Carolina State University
Campus Box 7535, Raleigh, NC 27695-7535 †
chirkova@csc.ncsu.edu
[3] Department of Archive and Library Sciences, Ionian University,
Palea Anaktora, Plateia Eleftherias, 49100 Corfu, Greece
manolis@ionio.gr

Abstract. The problem of optimizing queries in the presence of materialized views and the related view-design problem have recently attracted a lot of attention. Significant research results have been reported, and materialized views are increasingly used in query evaluation in commercial data-management systems. At the same time, most results in the literature assume set-theoretic semantics, whereas SQL queries have bag-theoretic semantics (duplicates are not eliminated unless explicitly requested). This paper presents results on selecting views to answer queries in relational databases under set, bag, and bag-set semantics. The results can be used under each of the three assumptions, to find sound and complete algorithms for designing views and rewriting queries efficiently.

Keywords: Data integration, query transformation, database theory.

1 Introduction

A lot of work has been done recently on optimizing queries in the presence of materialized views. In this context, problems such as definition of views, composition of views, maintenance of views have been researched. At the same time, the majority of the research assumes set-theoretic semantics, while SQL queries have bag-theoretic semantics, where duplicates are not eliminated unless explicitly requested. As SQL is the query language used in most commercial database-management systems (DBMS), results on rewriting queries under bag or bag-set semantics are useful in practice.

*** The project is co-funded by the European Social Fund (75 %) and National Resources (25%)-EPEAEK, Herakleitos.

† This author's work on this material has been supported by the National Science Foundation under Grant No. 0307072.

The problem of view selection has received significant attention in the literature [3, 4, 12–15, 17, 20, 21]. In this paper we consider view selection under set, bag, and bag-set semantics. The problem is as follows: Given a set of queries (which we call a *query workload*), a database, and a set of constraints on materialized views (e.g., *storage limit*, which is a bound on the amount of disk space available for storing the materialized views), return definitions of views that, when materialized in the database, would satisfy the constraints and reduce the evaluation costs of the queries. 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 [4, 15, 20]. Another motivation is provided by recent versions of several commercial DBMS, which support incremental updates of materialized views and use materialized views to speed up query evaluation. Choosing an appropriate set of views to materialize in the database is crucial in order to obtain performance benefits from these new features [3]. 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, such as in peer-based data management [11].

Database relations are often duplicate-free. More precisely, database relations are often sets, while views and queries are often bags, defined without using the DISTINCT keyword (bag-set semantics). The bag-set semantics case is arguably more practical than the bag-semantics case, as relational database-management systems typically compute query answers using operators with bag-valued outputs on set-valued databases. At the same time, studying the bag-semantics case is important not just from the theoretical but also from practical perspective, as in view selection it is possible to design and materialize bag-valued views and thus to obtain bag-valued databases of stored data. Computing query answers on such databases using the rules of evaluating SQL queries on relational databases obeys the laws of bag, rather than bag-set, semantics.

We now give examples that illustrate the high complexity of the problem of selecting views to materialize when set semantics are assumed. In this paper we show that under bag or bag-set semantics the complexity of the view-selection problem is significantly lower, and thus it is more likely to find efficient view-selection algorithms that output "more optimal" views than in the set-semantics case. In the following, we make a distinction between views that contribute to the construction of tuples in the query answers — we call them *containment-target views* — and optional *filtering views* [2, 19] that may improve the efficiency of query processing; see Example 2 for more details.

Our first example shows that the search space of potentially useful views can be very large even for simple and common select-project-join queries.

Example 1. We exhibit a workload of select-project-join queries and a storage limit, such that it is not possible to materialize the answers to all the workload queries. We consider for materialization select-project-join views, such that each view alone satisfies the storage limit and can support all workload queries. This example shows that for the given workload and under *set* semantics, at least an

exponential (in the size of the query workload) number of such views have more subgoals than any query in the input workload.[4]

Consider a large retail chain with multiple stores and warehouses, where products are ordered and shipped daily from the warehouses to replenish the inventory in the stores. Suppose that the database has a `Shipments` relation, and let `warehouseID, warehouseCity, storeID, storeCity`, `orderNumber`, `shipmentNumber`, and `shipmentDate` be some of its attributes. Suppose the employees of the retail chain contract shipments to independent truck drivers, by attracting them with tours connecting two or more cities. The company predefines a number of tour types to offer to the truck drivers, and the company employees 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 "two-city roundtrip": The query returns all cities `warehouseCity` such that there exist two scheduled deliveries: one from `warehouseCity` to some `storeCity` on a given `shipmentDate 'date1'`, and the other from (a warehouse in) `storeCity` back to (some store in) `warehouseCity` on a later `shipmentDate 'date2'`. We now give a SQL definition of the query; note the `DISTINCT` keyword that enforces set semantics.

```
SELECT DISTINCT S1.warehouseCity FROM Shipments S1, Shipments S2
WHERE S1.storeCity = S2.warehouseCity AND S1.warehouseCity = S2.storeCity
AND S1.shipmentDate = 'date1' AND S2.shipmentDate = 'date2';
```

As we show formally in the extended version of this paper, under set semantics, for a given storage limit and for a query workload that has several such tour queries for tours of different lengths, we can select and materialize a *single* view such that the number of copies of the `Shipments` relation in the `FROM` clause of the view is *exponential* in the combined size of the query workload. Thus, in view selection under set semantics we potentially need to consider up to an exponential number of views in the size of the input query workload. At the same time, as shown in this paper (see Theorems 4 and 7), in view selection under bag or bag-set semantics we do not need to consider views whose definitions have more subgoals than the number of subgoals of the longest input query.

Our second example shows that even if we further simplify the language of queries of interest, there still remains a large number of views that could significantly reduce the evaluation costs of the queries under set semantics.

Example 2. We use here the application domain of Example 1. Suppose that in addition to the `Shipments` relation described in Example 1, the database also has an `Inventory` relation, where the attributes of interest to us are `storeID, productID, orderNumber,` and `isOutstanding`. We assume that the volume of daily orders to replenish store inventories is large, and that a single order is typically made for a large range of product types and potentially for several stores in the same area. Further, suppose that different product types are stored

[4] This example is a variation on Example 1 in [8]; unlike that example, here we restrict the search space of views in that we do not consider filtering views.

at different warehouses, and thus all the products ordered in a single order can be delivered to a store via multiple shipments. Finally, we assume that on delivery, the contents of most — but not all — shipments are put on the store shelves and reflected in the inventory records on the same day.

Suppose that at the end of each day, the management of the retailer chain routinely runs certain "daily report" queries on the deliveries. One of the queries asks for the stores and product IDs such that some quantity of the product has been ordered and was to be delivered to the store on the day in question, but the order is still listed as "outstanding" in the store inventory. The query in SQL is:

```
SELECT DISTINCT I.storeID, productID FROM Shipments S, Inventory I
WHERE S.storeID = I.storeID and S.orderNumber = I.orderNumber
AND shipmentDate = 'today' AND isOutstanding = 'yes';
```

Note that unlike the queries in Example 1, the `FROM` clause of this query has just one copy of each relation. That is, the daily-report query *has no self-joins*. We show in this paper (see Sections 3.1, 4.1, and 5.1) that even under set semantics (as well as under bag and bag-set semantics), when doing view selection for such queries we do not need to consider views whose definitions have more subgoals than the input queries. At the same time, we still need to consider a significant number of views. Under the domain assumptions in this example, even using reasonable indexes on `Shipments` or `Inventory` does not eliminate most redundant tuples in the result of the join, and thus postprocessing of a large temporary join result (which by definition never even has an index) is part of any query plan that does not use materialized views. (The reason is, under our domain assumptions a shipment where `shipmentDate = 'today'` typically corresponds to a large number of product IDs in the `Inventory` relation, where the value of `isOutstanding` is `'no'` for most records. Conversely, for a large number of combinations of values of `storeID` and `productID` in the `Inventory` relation where `isOutstanding = 'yes'`, the value of `shipmentDate` of the corresponding order is not `'today'`.)

Consider materializing a view `V` that is a natural join of those tuples in `Shipments` and `Inventory` where `shipmentDate = 'today'` and `isOutstanding = 'yes'`. Suppose that in the answer to `V` we have (at least) attributes `storeID` and `orderNumber`. Then we can use this view as a *filter* — essentially like an index or a semijoin — to narrow down the number of tuples in the large result of joining the relations `Shipments` and `Inventory`. The query plans that use `V` all join `V` with either `Shipments` or `Inventory` first (an index can be maintained on `V` to make the join efficient), and then join the resulting *smaller-size* temporary relation with the remaining relation in the `FROM` clause of the query.[5]

We have seen that under set semantics, having a materialized view such as `V` could improve the processing efficiency of the daily-report query, which is important when the query is asked often and regularly on large relations.[6] At

[5] In presence of frequent updates on the stored data, the update problem for the view `V` is not much more complex than the problem of updating indexes on `Shipments` or `Inventory`, because we have just two relations in the `FROM` clause of the view.

[6] For instance, WalMart maintains a database with billions of rows in stored relations.

the same time, the choice of output attributes in a filtering view of this type depends on a number of criteria, including the types of other queries in the query mix. Thus, potentially we would have to consider *all* subsets of the combination of all attributes of the stored relations `Shipments` and `Inventory`.

We show in this paper that unlike the case of set semantics, filtering views do not need to be considered in view selection under bag or bag-set semantics. Thus, in view selection we can further restrict the search space of views that are useful in rewriting the given workload queries.

In this paper we present results on designing views to answer queries and on rewriting queries in relational databases under set, bag and bag-set semantics, which are useful in practice. The contributions are the following: (1) We give a bound for the number of subgoals in the optimal viewsets, and (2) we study the computational complexity of the view-selection problem. The results can be used in finding sound and complete algorithms for designing views and for rewriting queries under each of the three semantics.

To the best of our knowledge, limited related work has been done. [7] studies the containment problem of conjunctive queries under bag semantics which is proved to be $\prod_2^p$-hard, whereas equivalence under bag semantics has the same complexity as the graph-isomorphism problem, which is in NP. [10] presents techniques for bag semantics, bag-specific constraints (UWDs), and for handling bag queries over arbitrary mixes of bag and set schema elements and views. The problem of optimizing queries with materialized views under bag semantics is studied in [6] and under set semantics in [18]. Finally, [16] studies conjunctive queries over generalized databases, to obtain an understanding of the behavior of relations as multisets (cf. SQL query-evaluation semantics).

2 Preliminaries

2.1 Basic definitions

A *relational database* is a collection of stored relations. Each relation R is a collection of tuples, where each tuple is a list of values of the attributes in the *relation schema* of R. A relation can be viewed either as a *set* or as a *bag* (another term is *multiset*) of tuples. A bag can be thought of as a set of elements (we call it the *core-set* of the bag) with multiplicities attached to each element. In a *set-valued database*, all stored relations are sets; in a *bag-valued database*, multiset stored relations are allowed.

In this paper we focus on select-project-join SQL queries with equality comparisons, a.k.a. safe *conjunctive queries*. A conjunctive query is a rule of the form: $Q: \quad ans(\bar{X}) \leftarrow e_1(\bar{X}_1), \ldots, e_n(\bar{X}_n)$, where $e_1, \ldots e_n$ are database relations and $\bar{X}, \bar{X}_1, \ldots, \bar{X}_n$ are vectors of variables. The variables in $\bar{X}$ are called *head* or *distinguished variables* of Q, whereas the variables in $\bar{X}_i$ are called *body variables* of Q. A query has *self-joins* if the minimized query definition [5] has at least two subgoals with the same relation name. A *view* refers to a named query. A view is said to be *materialized* if its answer is stored in the database.

We say that a bag B is a *subbag* [7] of a bag B' (we write $B \subseteq_b B'$) if each element of B is contained in B' with a greater than or equal multiplicity. The *bag union* ($\sqcup$) [7] of two bags is obtained by adding the multiplicity factors for each tuple in each bag.

2.2 Query containment and equivalence

A query Q_1 is *set-contained* in a query Q_2, denoted by $Q_1 \sqsubseteq_s Q_2$, if for any set-valued database $\mathcal{D}$ the answer to Q_1 on $\mathcal{D}$ under set semantics is a subset of the answer to Q_2 on $\mathcal{D}$ under set semantics. A query Q_1 is *bag-contained* (*bag-set contained*) in Q_2, denoted by $Q_1 \sqsubseteq_b Q_2$ ($Q_1 \sqsubseteq_{bs} Q_2$, respectively), if for any bag-valued (set-valued, respectively) database $\mathcal{D}$, the answer to Q_1 on $\mathcal{D}$ under bag semantics (bag-set semantics, respectively) is a subbag of the answer to Q_2 on $\mathcal{D}$ under the same semantics. Two queries are *equivalent under set/bag/bag-set semantics* ($Q_1 \equiv_s Q_2$, $Q_1 \equiv_b Q_2$, $Q_1 \equiv_{bs} Q_2$, respectively) if they are contained in each other under that semantics.

2.3 Equivalent rewritings and the view-selection problem

Let $\mathcal{V}$ be a set of views defined on a database schema $\mathcal{S}$, and $\mathcal{D}$ be a database with the schema $\mathcal{S}$. Then by $\mathcal{D}_\mathcal{V}$ we denote the database obtained by computing all the view relations in $\mathcal{V}$ on $\mathcal{D}$. Let Q be a query defined on $\mathcal{S}$, and $\mathcal{V}$ be a set of views defined on $\mathcal{S}$. A query R is a *rewriting of the query Q using the views in $\mathcal{V}$* if all subgoals of R are view predicates defined in $\mathcal{V}$ or interpreted predicates.

The *expansion* R^{exp} of a rewriting R of a query Q using views is obtained from R by replacing all the view atoms in the body of R by their definitions in terms of the base relations. A rewriting R of a query Q on a set of views $\mathcal{V}$ is a *contained rewriting* of Q using $\mathcal{V}$ under set semantics if for every database $\mathcal{D}$, $R(\mathcal{D}_\mathcal{V}) \subseteq Q(\mathcal{D})$. A rewriting R of a query Q on a set of views $\mathcal{V}$ is an *equivalent rewriting* under set semantics if for every database $\mathcal{D}$, $R(\mathcal{D}_\mathcal{V}) = Q(\mathcal{D})$.

The definition of the notion *contained rewriting* for *bag* or *bag-set* semantics is analogous. The only difference is that we now require that $R(\mathcal{D}_\mathcal{V})$ is a subbag of $Q(\mathcal{D})$. The definition of the notion of *equivalent rewriting* for the *bag* and *bag-set* semantics is the same as above (in this case, however, the symbol '=' stands for bag equality). A conjunctive equivalent rewriting Q' of a conjunctive query Q (under some semantics) is *locally minimal* [18] if we cannot remove any literals from Q' and still retain equivalence to Q (under the same semantics).

Given a set, or *workload*, $\mathcal{Q}$ of queries on stored relations and a database instance, we want to find and precompute offline a set of intermediate results, defined as views (we call this set of views a *viewset*) on these relations. The views can be used to compute the answers to all queries in the workload $\mathcal{Q}$. Our goal is to design minimal-cost views, that is, views whose use in the rewriting of the queries in $\mathcal{Q}$ minimizes the evaluation cost of these queries. As we assume that the view relations have been precomputed and stored in the database, we do not assume any cost of computing the views. For query-evaluation costs, we use the following *sum-cost model* [8]. The cost of a join of two relations is proportional

to the sum of the sizes of the input and output relations.[7] The cost of a query plan is proportional to the sum of costs of all the joins in the plan. (We assume the use of left-linear query plans, where selections are pushed as far as they go and projection is the last operation.) The cost of evaluating a query is the minimum cost of its query plans. The total cost of evaluating a query workload is proportional to the sum of the costs of its queries; the sum can be weighted to reflect the relative frequency or importance of the queries.

In this paper we consider *problem inputs* that are 4-tuples $(\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, where $\mathcal{S}$ is a database schema, $\mathcal{Q}$ is a workload of queries defined on $\mathcal{S}$, $\mathcal{D}$ is a database with schema $\mathcal{S}$, and $\mathcal{L}$ is a collection of constraints on sets of materialized views. A problem input $\mathcal{P}$ is said to be *set-oriented* (*bag-oriented*, *bag-set-oriented*, respectively) if we consider set-semantics (bag-semantics, bag-set semantics, respectively) for computing query answers; $\mathcal{P}$ is said to be *conjunctive* if we consider the conjunctive language for queries, views and rewritings.

Some results in this paper are given for a special type of constraints $\mathcal{L}$ on materialized views: In those results, $\mathcal{L}$ is a singleton $\mathcal{L} = \{C\}$, $C \in \mathbf{N}$. The *storage limit* C means that the total size $size(\mathcal{V}(\mathcal{D}))$ of the relations for the views in $\mathcal{V}$ on $\mathcal{D}$ must not exceed C. If the storage limit is sufficiently large then we can materialize all query answers and this is the optimal viewset. The problem becomes interesting when the storage limit is less than that.[8]

Definition 1. *For a given query Q, semantics (set, bag, or bag-set) for evaluating the query on the database, a viewset $\mathcal{V}$, and database $\mathcal{D}$: (1) R is a* candidate rewriting *of Q in terms of $\mathcal{V}$ if R is an equivalent rewriting of Q under the given semantics, and (2) R is an* optimal rewriting *of Q in terms of $\mathcal{V}$ on $\mathcal{D}$ if R is a candidate rewriting that minimizes the cost of computing the answer to Q on $\mathcal{D}_\mathcal{V}$ among all candidate rewritings of Q in terms of $\mathcal{V}$.*

Definition 2. *Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be a problem input. A set of views $\mathcal{V}$ is said to be an* admissible viewset *for $\mathcal{P}$ if: (1) $\mathcal{V}$ gives equivalent (*candidate*) rewritings of all the queries in $\mathcal{Q}$, (2) for every view $V \in \mathcal{V}$, there exists an equivalent rewriting of a query in $\mathcal{Q}$ that uses V, and (3) $\mathcal{V}$ satisfies the constraints $\mathcal{L}$.*

Definition 3. *For a problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, an* optimal viewset *is a set of views $\mathcal{V}$ defined on $\mathcal{S}$, such that: (1) $\mathcal{V}$ is an admissible viewset for $\mathcal{P}$, and (2) $\mathcal{V}$ minimizes the total cost of evaluating the queries in $\mathcal{Q}$ on the database $\mathcal{D}_\mathcal{V}$, among all admissible viewsets for $\mathcal{P}$.*

Definition 4. *For any problem input $\mathcal{P}$, a viewset $\mathcal{V}$ is said to be* nonredundant *for $\mathcal{P}$ if $\mathcal{V}$ is admissible for $\mathcal{P}$ and there is no proper subset $\mathcal{V}'$ of $\mathcal{V}$ such that $\mathcal{V}'$ is also admissible for $\mathcal{P}$.*

In some results of this paper, instead of a database $\mathcal{D}$ in the definition of a problem input, we will use the notion of an *oracle* $\mathcal{O}$. An oracle is supposed to give, instantaneously, exact relation sizes for all views defined on the schema

[7] This models faithfully hash joins and index joins.

[8] If the storage limit is too small then there is no viewset that can rewrite all queries.

$\mathcal{S}$. In this case a problem input is written as $(\mathcal{S}, \mathcal{Q}, \mathcal{O}, \mathcal{L})$. The notion of an optimal viewset is defined analogously to the case of problem inputs of the form $(\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, where $\mathcal{D}$ is a database. The results in the remainder of this paper are given for problem inputs that include a fixed database, but can be extended in a straightforward manner to problem inputs that include an oracle.

2.4 Different types of views

There are two types of conjunctive views that can be used in a conjunctive rewriting of a conjunctive query [9]: (1) containment-target views, and (2) filtering views. A conjunctive view V is a *containment-target view* for a conjunctive query Q if there exists a conjunctive rewriting P of Q (P uses V), and there is a containment mapping (for the set-semantics case, or bijective mappings for the bag and bag-set semantics case) from Q to the expansion P^{exp} of P, such that V provides the image of at least one subgoal of Q under the mapping. Intuitively, in a rewriting, a *containment-target view* "covers" at least one query subgoal. Covering all query subgoals is enough to produce a rewriting of the query. A view is a *filtering view* for a query if it is not a containment-target view.

3 Queries without Self-Joins under Set Semantics

In this section we consider the view-selection problem under *set* semantics. We present the following results:

1. In Section 3.1 we show that for workloads of queries without self-joins there exist optimal viewsets whose view definitions do not have self-joins. Moreover, the view definitions in such viewsets have no more subgoals than any query in the workload.
2. In Section 3.2 we show that for workloads of queries without self-joins there exist optimal viewsets, such that rewriting any workload query does not require self-joins of containment-target views in the viewset.
3. In Section 3.3 we show that the decision version of the view-selection problem is in NP for workloads of queries without self-joins, provided that filtering views are not used in query rewritings.

These results are very useful in designing an algorithm that constructs optimal viewsets, as they provide a bound on the number of atoms in each view in an optimal viewset and ensure that these views do not contain self-joins, provided the workload queries do not contain self-joins.

3.1 View definitions without self-joins

The following theorem holds for queries without self-joins under set semantics.

Theorem 1. *Given a conjunctive set-oriented problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, where $\mathcal{L}$ represents a single storage limit and all queries in $\mathcal{Q}$ are conjunctive*

queries without self-joins, if there exists an optimal viewset $\mathcal{V}$ for $\mathcal{P}$ under the storage limit constraint $\mathcal{L}$, then there exists an optimal viewset $\mathcal{V}'$ under $\mathcal{L}$ such that each view in $\mathcal{V}'$ can be defined as a conjunctive query without self-joins.

The statement of Theorem 1 is that whenever workload queries have no self-joins then there exist optimal viewsets whose view definitions do not have self-joins. The following Corollary 1 goes one step further, by showing that each view in the optimal viewset has no more subgoals than the workload queries.

Corollary 1. *Given a conjunctive set-oriented problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ where $\mathcal{L}$ represents a single storage-limit constraint and assuming that (1) all queries in $\mathcal{Q}$ are without self-joins, and (2) the number of (relational) subgoals in any query does not exceed an integer n, then if there exists an optimal viewset for $\mathcal{P}$ under $\mathcal{L}$, then there exists an optimal viewset $\mathcal{V}$ of $\mathcal{P}$ under $\mathcal{L}$, such that for every view V in $\mathcal{V}$, the number of subgoals of V is bounded from above by n.*

The optimal viewset stipulated in Corollary 1 may include filtering views alongside containment-target views. Moreover, even an exponential number of filtering views may be necessary under set semantics; see [8]. Note that we cannot strengthen the Corollary 1 to state that under the premises of the corollary there exists an optimal viewset $\mathcal{V}$, such that each view in $\mathcal{V}$ is a subexpression of some query in the input query workload. As a counterexample, consider Example 3.

Example 3. Consider a query workload $\mathcal{Q}$ = { `Q1, Q2` }, where:

```
Q1(X,Y,Z) :- p(X,X), s(X,Y), t(Y,Z).
Q2(X,Y,Z) :- p(X,Y), s(Y,Y), t(Y,Z).
```

Suppose that we are given a database $\mathcal{D}$= { `p(a,a)`, `p(a,b)`, `p(c,c)`, `s(a,a)`, `s(b,b)`, `s(c,b)`, `t(a,d)`, `t(a,f)`, `t(b,g)`, `t(b,h)`} and a set of constraints $\mathcal{L}$ = { L }, where the value of the storage limit $L = 6$ is an upper bound on the sizes of materialized views on $\mathcal{D}$. Consider a view V:

```
V:  v(Z,T,W,U) :- p(Z,T),s(T,W),t(W,U).
```

Note that the view V is not a subexpression of either query in the workload $\mathcal{Q}$. However, $\mathcal{V}$ = { V } is an optimal viewset for the problem input ({ $P(A,B)$, $S(C,D)$, $T(E,F)$ }, { `Q1, Q2` }, $\mathcal{D}$, { L }), and both input queries can be rewritten using the view V:

```
Q1'(X,Y,Z):- v(X,X,Y,Z).
Q2'(X,Y,Z):- v(X,Y,Y,Z).
```

Finally, when queries have self-joins, the number of subgoals of views can be up to a product of the number of subgoals of the queries; see Example 1 in [8].

3.2 Rewritings without self-joins

While the results in Section 3.1 refer to the stucture of the views in an optimal viewset, in this section we are interested in the structure of the query rewritings provided that the queries in the workload $\mathcal{Q}$ do not have self-joins. We show that there exists an optimal viewset $\mathcal{V}$, such that rewriting any query in $\mathcal{Q}$ does not require self-joins of containment-target views in $\mathcal{V}$.

Theorem 2. *Given a conjunctive set-oriented problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ and assuming that the queries in $\mathcal{Q}$ do not have self-joins, if there exists an optimal viewset $\mathcal{V}$ for $\mathcal{P}$ under the storage limit $\mathcal{L}$, then it is possible to rewrite each query in $\mathcal{Q}$ using $\mathcal{V}$, without self-joins of containment-target views.*

The lack of self-joins in queries is an essential condition in Theorem 2, as we can show that otherwise nontrivial self-joins of containment-target views may be required. An analogous result holds for filtering views (cf. Example 1 in [8]).

3.3 Complexity

We now consider the decision version of the view-selection problem, that is, given a set-oriented problem input $\mathcal{P}$ and a positive integer K, the problem is to determine whether there exists an admissible viewset $\mathcal{V}$ for $\mathcal{P}$, such that the cost of evaluating the queries $\mathcal{Q}$ in $\mathcal{P}$ using $\mathcal{V}$ does not exceed K. We show that this problem is in NP for workloads of queries without self-joins, provided that filtering views are not used in query rewritings. We prove the result for the *oracle* version of problem inputs, that is, we show that the size of a witness is polynomial in the size of the following components of the problem input: database schema, query workload, and constraints on the materialized views. This result is stronger than proving that the size of a witness is polynomial in the size of the above components plus the size of an input database, because database schemas, query workloads, and constraints on the materialized views are typically small in size compared to the size of possible databases conforming to the schemas. To prove the main result, we first establish an upper bound on the number of containment-target views in query rewritings.

Lemma 1. *[18] Let Q be a conjunctive query and $\mathcal{V}$ be a set of views, both Q and $\mathcal{V}$ without built-in predicates. If the body of Q has p relational subgoals and Q' is a locally minimal equivalent conjunctive rewriting of Q using $\mathcal{V}$, then Q' has at most p relational subgoals.*

Proposition 1. *For any conjunctive query Q with p relational subgoals and for any locally minimal conjunctive rewriting Q' of Q in terms of views such that $Q' \equiv_s Q$, the number of containment-target views in Q' does not exceed p.*

This result follows from the observation that any locally minimal rewriting does not contain filtering views. From Proposition 1 we obtain Theorem 3:

Theorem 3. *Given an* oracle version *of a conjunctive set-oriented problem input $\mathcal{P}$ whose queries are without self-joins, the decision version of the view-selection problem is in NP, provided that rewritings do not include filtering views.*

Note that if filtering views are allowed in query rewritings, then the view-selection problem under set semantics has an exponential-time lower bound even when none of the workload queries have self-joins; see [8].

4 Queries under Bag Semantics

In this section we consider the view-selection problem under bag semantics. Before proceeding to the main results of this section, we note that under bag semantics any candidate query rewriting lacks any filtering views, as well as any redundant containment-target views [7]. We now summarize the main results:

1. In Section 4.1 we show that for workloads of queries with or without self-joins, each view definition in any admissible viewset (and thus in any optimal viewset) has no more subgoals than any query in the input workload. As a consequence, for workloads of queries without self-joins each view definition in an admissible viewset can be defined without self-joins.
2. In Section 4.2 we show that for workloads of queries without self-joins and for any admissible viewset, rewriting any query in the workload does not require self-joins of view atoms.
3. In Section 4.3 we show that the decision version of the view-selection problem is in NP for workloads of queries with or without self-joins and for a single storage-limit constraint on materialized views (see also [1]).

Comparing these results with those in Section 3, we conclude that both constructing admissible/optimal viewsets and rewriting queries using views are easier problems under bag semantics than under set semantics.

4.1 Bounded number of subgoals

The following lemma holds for workloads of queries without *or with* self-joins under bag semantics and for arbitrary sets of constraints on materialized views.

Lemma 2. *Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be a conjunctive bag-oriented problem input, where $\mathcal{L}$ is a set of* any *constraints. Let $\mathcal{V}$ be any* admissible *viewset for $\mathcal{P}$, and let Q be any query in $\mathcal{Q}$. Suppose $\mathcal{V}' \subseteq \mathcal{V}$ is the set of all views in the equivalent rewriting R of Q in terms of $\mathcal{V}$. Then the definitions of views $\mathcal{V}'$ in the expansion of R form a partition of the definition of Q.*

The remaining results in Section 4.1 follow trivially from Lemma 2.

Proposition 2. *Given a conjunctive bag-oriented problem input $\mathcal{P}$, let $\mathcal{V}$ be any admissible viewset for $\mathcal{P}$. Then each view in $\mathcal{V}$ has at most n subgoals, where n is the number of subgoals in the longest query in the input workload $\mathcal{Q}$.*

The following theorem sets an upper bound on the number of subgoals in the body of any view definition in any admissible viewset.

Theorem 4. *Let $\mathcal{P}$ be a conjunctive bag-oriented problem input and n be the number of subgoals in the longest query in $\mathcal{Q}$. Then, for any admissible viewset $\mathcal{V}$ for $\mathcal{P}$, each view in $\mathcal{V}$ can be defined using at most n subgoals.*

We can make a more precise statement about the number of subgoals in view definitions for views in admissible viewsets: Let $\mathcal{P}$ be a conjunctive bag-oriented problem input and let V be any view in any *admissible* viewset $\mathcal{V}$ for $\mathcal{P}$. Suppose V is used in rewriting queries $Q_{i_1}, \ldots, Q_{i_k}$ in $\mathcal{Q}$; let m be the number of subgoals in the *shortest* definition among $Q_{i_1}, \ldots, Q_{i_k}$. Then V can be defined using at most m subgoals. In addition, we observe the following. For any conjunctive bag-oriented problem input $\mathcal{P}$ and for an admissible viewset $\mathcal{V}$ for $\mathcal{P}$: If queries in $\mathcal{Q}$ do not have self-joins, then every view in $\mathcal{V}$ can be defined as a conjunctive query without self-joins.

4.2 Rewritings without self-joins of views

Analogously to the case of set semantics, in the case of bag semantics we can show that for problem inputs $\mathcal{P}$ whose query workloads $\mathcal{Q}$ do not have self-joins, and for any admissible viewset $\mathcal{V}$ for $\mathcal{P}$, rewriting any query in $\mathcal{Q}$ does not require self-joins of views in $\mathcal{V}$. The following theorem follows directly from Lemma 2.

Theorem 5. *Let $\mathcal{P}$ be a conjunctive bag-oriented problem input and $\mathcal{V}$ an admissible viewset for $\mathcal{P}$. Assuming that queries in $\mathcal{Q}$ do not have self-joins, then it is possible to rewrite each query in $\mathcal{Q}$ without using self-joins of views in $\mathcal{V}$.*

4.3 Complexity

In this section we show that the decision version of the view-selection problem is in NP for a single storage-limit constraint on materialized views (see also [1]). We define the decision version of the problem and state the result for the *oracle* version of problem inputs, analogously to the respective formulations in Section 3.3. At the same time, unlike the results in Section 3.3, the NP results for bag semantics hold for workloads of queries without *or with* self-joins.

We first establish an analog of Proposition 1 in Section 3.3:

Proposition 3. *For any conjunctive query Q with p relational subgoals and for any conjunctive rewriting Q' of Q in terms of views, such that $Q' \equiv_b Q$, the number of views in Q' does not exceed p.*

This result follows immediately from the fact that for any equivalent (under bag semantics) rewriting to a query, the rewriting does not contain filtering views or "unnecessary" containment-target views, and is thus locally minimal.

We now establish Theorem 6, which is a direct consequence of the following observation: Under bag semantics, for any problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ with

any set of constraints $\mathcal{L}$, and for any admissible viewset $\mathcal{V}$ for $\mathcal{P}$, the number of views in $\mathcal{V}$ does not exceed p, where p is the total number of relational subgoals in all the queries in the query workload $\mathcal{Q}$ in $\mathcal{P}$.

Theorem 6. *Given an* oracle version *of a conjunctive, bag-oriented problem input $\mathcal{P}$ and assuming that the input set of constraints $\mathcal{L}$ represents a single storage limit, the decision version of the view-selection problem is in NP.*

5 Queries under Bag-Set Semantics

In this section we consider the view-selection problem under bag-set semantics. Note that all queries mentioned in the results below may have self-joins. In the extended version of the paper we show that filtering views are not needed under bag-set semantics. The main results of this section are:

1. The results in Section 5.1 are similar to those in Section 4.1, which concerns bag semantics. That is, we show that for workloads of queries with or without self-joins, each view definition in any admissible viewset (and thus in any optimal viewset) has no more subgoals than any query in the input workload. As a consequence, for workloads of queries *without* self-joins, each view definition in an admissible viewset can be defined without self-joins. Moreover, we can show that for workloads of queries without self-joins and for any admissible viewsets, rewriting any query in the workload does not require self-joins of view atoms.
2. In Section 5.2 we show that the decision version of the view-selection problem is in NP for workloads of queries with or without self-joins and for a single storage-limit constraint on materialized views.

5.1 Bounded number of subgoals

Our first result holds for workloads of queries without *or with* self-joins under bag-set semantics and for arbitrary sets of constraints on materialized views.

Lemma 3. *Let $\mathcal{P}$ be a conjunctive, bag-set-oriented problem input, and let $\mathcal{V}$ be any* admissible *viewset for $\mathcal{P}$. Then each view in $\mathcal{V}$ has at most n subgoals, where n is the number of subgoals in the longest query in the input workload $\mathcal{Q}$.*

All remaining results in this subsection follow directly from Lemma 3.

Theorem 7. *Let $\mathcal{P}$ be a conjunctive bag-set-oriented problem input, and n be the number of subgoals in the longest query in $\mathcal{Q}$. Then, for all admissible viewsets $\mathcal{V}$ for $\mathcal{P}$, each view in each $\mathcal{V}$ can be defined using at most n subgoals.*

We can make a more precise statement about the number of subgoals in view definitions for views in admissible viewsets:

Corollary 2. *Let $\mathcal{P}$ be a conjunctive bag-set-oriented problem input, and let V be any view in any* admissible *viewset $\mathcal{V}$ for $\mathcal{P}$. Suppose V is used in rewriting queries $Q_{i_1}, \ldots, Q_{i_k}$ in $\mathcal{Q}$; let m be the number of subgoals in the* shortest *definition among the definitions of $Q_{i_1}, \ldots, Q_{i_k}$. Then V can be defined using at most m subgoals.*

We also make the following observation: Let $\mathcal{P}$ be a conjunctive bag-set-oriented problem input, and $\mathcal{V}$ an admissible viewset for $\mathcal{P}$. If queries in $\mathcal{Q}$ do not have self-joins, then each view in $\mathcal{V}$ can be defined as a conjunctive query without self-joins. In addition, we can show that for problem inputs $\mathcal{P}$ whose query workloads $\mathcal{Q}$ do not have self-joins and for any admissible viewset $\mathcal{V}$ for $\mathcal{P}$, rewriting any query in $\mathcal{Q}$ does not require self-joins of views in $\mathcal{V}$.

5.2 Complexity

We now show that the decision version of the view-selection problem is in NP for a single storage-limit constraint on materialized views. We formulate the decision version of the problem and state the result for the *oracle* version of problem inputs, similarly to the previous sections. The NP results hold for workloads of queries without *or with* self-joins.

Proposition 4. *For any conjunctive query Q with p relational subgoals and for any conjunctive locally minimal rewriting Q' of Q in terms of views, such that $Q' \equiv_{bs} Q$, the number of views in Q' does not exceed p.*

This result follows from the definition of a locally minimal rewriting that is equivalent to a query under bag-set semantics. By definition, the rewriting does not contain filtering views or "unnecessary" containment-target views.

We now establish that the decision version of the view-selection problem is in NP. This result is a consequence of the following observation: Under bag-set semantics, for any problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ with any set of constraints $\mathcal{L}$, and for any nonredundant viewset $\mathcal{V}$ for $\mathcal{P}$, the number of views in $\mathcal{V}$ does not exceed p, where p is the total number of relational subgoals in all the queries in the query workload $\mathcal{Q}$ in $\mathcal{P}$.

Theorem 8. *Given an* oracle version *of a conjunctive bag-set-oriented problem input $\mathcal{P}$ and assuming that the input set of constraints $\mathcal{L}$ represents a single storage limit, the decision version of the view-selection problem is in NP.*

6 Conclusions and Future Work

This paper presents results on designing views to answer queries in relational databases under set, bag and bag-set semantics. The results can be used in finding sound and complete algorithms for designing views and rewriting queries under each of the three assumptions. We are currently working on designing such algorithms. In our future work we also plan to study the complexity of the optimization problem and to extend this method to include, in a systematic way, queries with arithmetic comparisons. Applying these results to XQuery is another direction of our future research.

References

1. F. Afrati and R. Chirkova. Selecting and using views to compute aggregate queries. In *Proceedings of ICDT*, 2005.
2. F. Afrati, C. Li, and J.D. Ullman. Generating efficient plans for queries using views. In *Proceedings of ACM SIGMOD*, 2001.
3. S. Agrawal, S. Chaudhuri, and V.R. Narasayya. Automated selection of materialized views and indexes in SQL databases. In *Proc. VLDB*, pages 496–505, 2000.
4. E. Baralis, S. Paraboschi, and E. Teniente. Materialized view selection in a multidimensional database. In *Proceedings of VLDB*, pages 156–165, 1997.
5. A. K. Chandra and P. M. Merlin. Optimal implementation of conjunctive queries in relational databases. In *Proc. 9th ACM STOC*, pages 77–90, 1977.
6. S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim. Optimizing queries with materialized views. In *Proceedings of ICDE*, pages 190–200, 1995.
7. S. Chaudhuri and M. Y. Vardi. Optimization of real conjunctive queries. In *Proceedings of PODS*, pages 59–70. ACM Press, 1993.
8. R. Chirkova, A. Y. Halevy, and D. Suciu. A formal perspective on the view selection problem. *The VLDB Journal*, 11(3):216–237, 2002.
9. R. Chirkova and C. Li. Materializing views with minimal size to answer queries. In *Proceedings of PODS*, pages 38–48. ACM Press, 2003.
10. A. Deutsch. *XML Query Reformulation over Mixed and Redundant Storage*. PhD thesis, University of Pennsylvania, 2002.
11. S. Gribble, A. Halevy, Z. Ives, M. Rodrig, and D. Suciu. What can databases do for peer-to-peer? In *Proceedings of WebDB*, 2001.
12. H. Gupta. Selection of views to materialize in a data warehouse. In *Proceedings of ICDT*, pages 98–112, 1997.
13. H. Gupta, V. Harinarayan, A. Rajaraman, and J. D. Ullman. Index selection for OLAP. In *Proceedings of ICDE*, pages 208–219, 1997.
14. H. Gupta and I. S. Mumick. Selection of views to materialize under a maintenance cost constraint. In *Proceedings of ICDT*, pages 453–470, 1999.
15. V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing data cubes efficiently. In *Proceedings of ACM SIGMOD*, pages 205–216, 1996.
16. Y. Ioannidis and R. Ramakrishnan. Containment of conjunctive queries: Beyond relations as sets. *ACM Transactions on Database Systems*, 20(3):288–324, 1995.
17. H. J. Karloff and M. Mihail. On the complexity of the view-selection problem. In *Proceedings of PODS*, pages 167–173, Philadelphia, Pennsylvania, 1999.
18. A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava. Answering queries using views. In *Proceedings of PODS*, pages 95–104. ACM Press, 1995.
19. R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *Proceedings of VLDB*, pages 484–495, 2000.
20. D. Theodoratos and T. Sellis. Data warehouse configuration. In *Proceedings of VLDB*, pages 126–135, Athens, Greece, 1997.
21. J. Yang, K. Karlapalem, and Q. Li. Algorithms for materialized view design in data warehousing environment. In *Proceedings of VLDB*, pages 136–145, 1997.