On Solving Efficiently the View Selection Problem under Bag-Semantics*

Foto Afrati[1], Matthew Damigos[1], and Manolis Gergatsoulis[2]

[1] Department of Electrical and Computing Engineering,
National Technical University of Athens (NTUA), 15773 Athens, Greece
{afrati,mgdamig}@softlab.ntua.gr
[2] Department of Archive and Library Sciences, Ionian University,
Ioannou Theotoki 72, 49100 Corfu, Greece
manolis@ionio.gr

Abstract. In this paper, we investigate the problem of view selection for workloads of conjunctive queries under bag semantics. In particular we aim to limit the search space of candidate viewsets. In that respect we start delineating the boundary between query workloads for which certain restricted search spaces suffice. They suffice in the sense that they do not compromise optimality in that they contain at least one of the optimal solutions. We start with the general case, where we give a tight condition that candidate views can satisfy and still the search space (thus limited) does contain at least one optimal solution. Preliminary experiments show that this reduces the size of the search space significantly. Then we study special cases. We show that for chain query workloads, taking only chain views may miss all optimum solutions, whereas, if we further limit the queries to be path queries (i.e., chain queries over a single binary relation), then path views suffice. This last result shows that in the case of path queries, taking query subexpressions suffice.

1 Introduction

The *view selection problem* has received significant attention in many data-management scenarios, such as information integration, data warehousing, website designs, and query optimization. The static version of this problem is to choose a set of views to materialize over a database schema, such that (a) the cost of evaluating a set of queries is minimized, and (b) the views fit into a prespecified storage space. In query optimization, evaluating a set of queries using previously materialized views can significantly speed-up query processing, as part of the computation necessary for each query may have been done while computing views. Moreover, a set of similar queries (e.g. queries with similar

* This paper is part of the 03EΔ176 research project, implemented within the framework of the "Reinforcement Programme of Human Research Manpower" (PENED) and co-financed by National and Community Funds (25% from the Greek Ministry of Development-General Secretariat of Research and Technology and 75% from E.U.-European Social Fund).

M. Castellanos, U. Dayal, and T. Sellis (Eds.): BIRTE 2008, LNBIP 27, pp. 12–28, 2009.

subexpressions) can be computed efficiently by selecting an appropriate set of views that exploits these sharing opportunities. In a data warehouse, a successful selection of views to materialize can preclude costly access to the base relations and consequently helps to answer a batch of queries in efficient way. Similarly, the choice of a proper set of views to precompute may improve the performance of web-sites; because the set of expected queries can be answered quickly [10].

In contrast to the query answering problem using views, where the set of views is initially given, the view selection problem indicates automated techniques to produce the appropriate set of materialized views. In this paper, we focus on the view selection problem using query rewriting techniques, assuming that both query and view definitions are conjunctive queries. We use bag-semantics, which means that duplicate occurrences of tuples are allowed to query answers and to database relations [19]. The "bag-approach" of the problem is more practical because of its close relationship to the SQL features where bag-relations are allowed and the duplicate tuples are not eliminated during the query evaluation; unless explicitly requested (by using the DISTINCT keyword).

The hardness of the view selection, as defined and investigated in [8,2,7,12], is caused by the bicriteria nature of the problem. These criteria are: (1) for a given set of views, the selection of the less-costly equivalent rewritings of the queries and (2) the choice of the appropriate set of views which does not violate the storage constraint. Bicriteria settings have different variants (and consequently different solutions and complexity results) depending of which of the two objective functions of these two criteria is required to be optimized under the constraint that the value of the other objective function does not exceed a bound that is given by the designer of the system. In this paper, we consider the variant of the problem in which we want to find a viewset such that it does not exceed the storage constraint and is optimum with respect to the evaluation cost of the query workload. We count the size of a viewset as the number of tuples required to store all the views in the viewset (however we notice that all our results hold under a more general count) and the cost of query evaluation is based on the sum-of-joins cost model for left-linear query plans (the exact definitions can be found in subsequent section). In [2] the same problem is investigated and is shown that we can restrict the search space for views in the viewset only to those views that are *generalizations of query subexpressions*.

Our contributions in this paper are: a) In Section 4.1, we improve the search space of [2] by showing that it suffices to consider only the *least general generalizations of query subexpressions*. In particular, we show that if we restrict ourselves to this smaller search space, an optimal solution is always retained, i.e., a solution which satisfies the storage limit and achieves the optimum value for the evaluation cost. b) Based on these results, we develop in Section 4.2 an efficient algorithm for finding an optimal solution. c) We study (Section 5) the problem for two special cases, namely when all queries in the workload are chain queries and when they are path queries. In the first case we show that we cannot restrict the search space to only chain views because we may loose all optimal

solutions (Section 5.1). In the second case, we show that we can restrict further the search space to consider only path views (Section 5.2).

2 Related Work

The problem of automatic selection of views to materialize has attracted the interest of many researchers. In [7], the space requirements for the view selection problem in the context of data warehouse design under set-semantics, are considered. This paper, also investigates conditions under which the search space of optimal configurations can be reduced to the views that are subexpressions of the queries in the workload. In [21,23], the extraction of common subexpressions of the queries in the workload is studied. The authors in [21], study the problem of searching for a maximum common subexpression of a workload, while [23] proposes an algorithm for searching for maximum common subexpressions for a subclass of select-project-join SQL queries, using query graphs. Another approach for finding similar subexpressions is proposed in [25] where workloads of select-project-join-groupby queries are considered. The authors propose a solution for the multi-query optimization problem which is incorporated in the Microsoft SQL Server. The algorithm has a lightweight mechanism (table-signatures) to detect common subexpressions and multiple sharing opportunities.

In [12], it is stated the view selection problem using AND-OR graphs to represent the query plans. Two types of constraints on materialized views are assumed, a storage limit and a maintenance-cost constraint. The candidate set of view configurations are given as input, hence the time of the construction of view configurations is not considered in the response time of the algorithms.

In [20], the view selection problem assuming a maintenance-cost constraint in the data warehouse environment and proposed an algorithm based on multi-query graphs, is studied. In [24], the authors examine greedy/heuristic algorithms for solving the view-selection problem assuming a maintenance-cost constraint and OLAP queries in multidimensional data warehouse environment. In [6] the problem for multidimensional databases is studied and an algorithm that selects views by reducing significantly the solution space is proposed; considering only the relevant elements of the multidimensional lattice. The authors considered the standard SQL notion of group-by and aggregate functions in order to capture queries with aggregation. In earlier work, Rizzi and Saltarelli [18] presented a comparative evaluation that uses view materialization and indexing for a single GSPJ (Group-by-Select-Project-Join) query expressed on a star scheme for the data warehousing context.

The view selection problem, in the context of multidimensional data warehouses, also studied by several authors [14,13,11]. In [14], it is described a system which was incorporated in Microsoft SQL Server and focuses on selection of both views and indexes. Earlier, the authors of [13] propose algorithms for selecting views in the case of data cubes and study the complexity of the problem. In [11], the work of [13] was further extended to include index selection.

A significant result that underlines the difference of the view selection problem in the case of queries with and without aggregation is presented in [3]. In this

work, an algorithm for selecting views is proposed and complexity results are presented, using a theoretical approach to express GSPJ queries. The authors also showed that using materialized views to compute aggregate queries results greater benefits than for purely conjunctive queries; as a view with aggregation precomputes some of the grouping/aggregation on some of the query's subgoals.

In [8], Chirkova et al. observed that the complexity of view selection problem under set semantics, and assuming conjunctive query workload, depends crucially on the quality of the estimates that a query optimizer has on the size of views. In [8], it is also shown that an optimal choice of views may involve an exponential number of views in the size of the database schema. In the same context, in [2], Afrati et al. study the search space of candidate sets of views, under bag, set and bag-set semantics. Finally, the problem of selecting minimal-size-views to materialize has been studied theoretically in [9], where the problem has been proven decidable and an upper bound is given on this problem's complexity.

3 Preliminaries

3.1 Basic Definitions

A *relation schema* is a named relation defined by its name R (called *relation name*) and a set A of *attributes*. A *relation instance* r for a relation schema is a collection of tuples over its attribute set. The schemas of the relations in a database constitute its *database schema*. A *relational database instance* (*database*, for short) is a collection of stored relation instances. A relation instance can be viewed either as a *set* or as a *bag* (or *multiset*) of tuples. A *bag* (or bag-relation [22]) can be thought of as a set of elements 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. The *bag-operators* [22] are similar to the set-operators. The difference is that in bag-selection and bag-projection duplicate tuples in the result are not eliminated. Concerning the Cartesian product, the difference is that the multiplicity of each tuple t obtained in $R \times S$ from a tuple t_1 of R and a tuple t_2 of S is $m \cdot n$, where m is the multiplicity of t_1 and n is the multiplicity of t_2. Depending on whether a database is bag or set-valued and the operators are set or bag operators, the queries may be computed under *set-semantics* (considering set-valued databases and operators), *bag-semantics* (considering bag-valued databases and operators), or *bag-set semantics* (considering set-valued databases and bag-operators). We consider bag-semantics in this paper.

A *query* is a mapping from databases to databases, usually specified by a logical formula on the schema $\mathcal{S}$ of the input databases. Typically, the output database (called *query answer*) is a database with a single relation. In this paper we focus on the class of select-project-join SQL queries with equality comparisons, a.k.a. safe *conjunctive queries* (CQs for short). Formally, a conjunctive query definition [1] is a rule of the form:

$$Q : q(\overline{X}) \text{ :- } g_1(\overline{X}_1), \ldots, g_n(\overline{X}_n)$$

where $g_1, \ldots, g_n$ are database relations and $\overline{X}$, $\overline{X}_1$, ..., $\overline{X}_n$ are vectors of variables or constants. The atom $q(\overline{X})$ is the *head* of Q while the atoms on the right of :- are said to be the *body* of Q. Each $g_i(\overline{X}_i)$ is also called a *subgoal* of Q. The variables in $\overline{X}$ are called *distinguished* or *head variables* of Q, whereas the variables in $\overline{X}_i$ are called *body variables* of Q. A body variable which is not also a head variable is called *non-distinguished variable* of Q. In this work, we consider *safe conjunctive queries* that is CQs whose head variables also occur in their body. A *chain query* is a conjunctive query of the following form:

$$Q : q(X_0, X_n) \text{ :- } r_1(X_0, X_1), r_2(X_1, X_2), \ldots, r_n(X_{n-1}, X_n)$$

where $r_1, \ldots, r_n$, are binary relations and $X_0, X_1, \ldots, X_n$ are variables. If the relation symbols r_1, ..., r_n are identical then the query is called *path query* of length n, denoted as P_n. A *view* refers to a named query. A view is said to be *materialized* if its answer is stored in the database. In this work, we are restricted to the use of views defined by conjunctive queries called *conjunctive views*.

A *substitution* θ [15] is a finite set of the form $\{X_1/Y_1, \ldots, X_n/Y_n\}$, where each Y_i is a variable or a constant, and X_1, ..., X_n are distinct variables. When $Y_1, \ldots, Y_n$ are distinct variables, θ is called *renaming substitution*. In the following we also use the notion of *expression* to denote a conjunction of atoms. Let $\theta = \{X_1/Y_1, \ldots, X_n/Y_n\}$ be a substitution. Then the *instance* $E\theta$ of an expression (resp. a query) E, is the expression (resp. the query) obtained by simultaneously replacing each occurrence of X_i in E by Y_i for all $i = 1, \ldots, n$.

Definition 1. *An expression E is a* generalization *of an expression E' if E' is an instance of E. E is a* common generalization *of E_1, ..., E_n, with $n > 1$ if E is a generalization of each expression E_i, with $1 \leq i \leq n$. E is a* least common generalization *(or a* least general generalization - *lgg [16]) of E_1, ..., E_n, with $n > 1$, if E is a common generalization of E_1, ..., E_n, and for each common generalization G of E_1, ..., E_n, the expression G is a generalization of E.*

3.2 Query Rewriting and the View Selection Problem

Given a set of views (also, called *viewset*) $\mathcal{V}$ defined on a database schema $\mathcal{S}$, and a database $\mathcal{D}$ on the schema $\mathcal{S}$, then by $\mathcal{V}(\mathcal{D})$ we denote the database obtained by computing all the view relations in $\mathcal{V}$ on $\mathcal{D}$. Moreover, let Q be a query 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 atoms defined in $\mathcal{V}$. The *expansion* R^{exp} of R is obtained by replacing all view atoms in the body of R with their corresponding base relations. Non-distinguished variables in a view definition are replaced with fresh variables in R^{exp}. A rewriting R of a query Q on a viewset $\mathcal{V}$ is an *equivalent rewriting* if $R(\mathcal{V}(\mathcal{D})) = Q(\mathcal{D})$, for every database $\mathcal{D}$. In [19], it is proved that a rewriting R of a query Q, under bag-semantics, is equivalent to Q if and only if there is an one-to-one containment mapping from Q to the R^{exp}.

Given a set $\mathcal{Q}$ of queries (also called *query workload*), defined on a schema $\mathcal{S}$, and a database instance $\mathcal{D}$, we want to find and precompute offline a viewset

$\mathcal{V}$ defined on $\mathcal{S}$, such that the views in $\mathcal{V}$ can be used to compute the answers to all queries in the workload $\mathcal{Q}$. More specifically, our problem, called the *view selection problem*, is to find a set of views that when materialized, (a) would satisfy a set $\mathcal{L}$ of constraints on the size of the views, and (b) can be used to get equivalent rewritings of the queries in $\mathcal{Q}$ which minimizes the evaluation cost of the queries. We refer to the tuple $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ as the *input of view selection problem*. The view selection problem is said to be *bag-oriented* (resp. *set-oriented*, or *bag-set-oriented*) if we consider bag semantics (resp. set semantics, or bag-set semantics).

In this paper, we consider that the only constraint on materialized views is a storage limit L (i.e. $\mathcal{L} = \{L\}$), which is a bound on the size of the views (which represents the available disk space for storing the views). Our goal is to choose the viewsets which minimize the evaluation cost of the queries and whose size will not exceed the limit L. Notice that, if the storage limit is sufficiently large then we can materialize all query answers, which is an optimal viewset. The problem becomes interesting when the storage limit is less than that. In the following we measure the size of a relation R as the number of tuples in R.

Definition 2. *Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be a view selection problem input. A viewset $\mathcal{V}$ is said to be* admissible *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 at least one equivalent rewriting of a query in $\mathcal{Q}$ that uses V, and (3) $\mathcal{V}$ satisfies the constraints $\mathcal{L}$.*

The following definition formally defines the *solution* and *optimal solution* of view selection problem for a given input.

Definition 3. *Let a view selection problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$.*

- *A* solution *of $\mathcal{P}$ is a tuple $(\mathcal{V}_{adm}, \mathcal{R})$, where $\mathcal{V}_{adm}$ is an admissible viewset for $\mathcal{P}$ and $\mathcal{R}$ is a set of equivalent rewritings of the queries in $\mathcal{Q}$ using $\mathcal{V}_{adm}$.*
- *An* optimal solution *for $\mathcal{P}$ is a solution which minimizes the cost of evaluating the queries in the workload among all solutions of $\mathcal{P}$. The viewset in an optimal solution is said to be an* optimal viewset.

Optimal solutions relate to the estimation of the cost of evaluating a query. We thus demand from the optimal solutions to minimize a given cost-function that we employ. We assume that the view relations have been precomputed, hence we do not assume any cost of computing the views. For conjunctive queries we use the *sum-of-joins* cost model which measures the cost of query evaluation as the sum of the costs of all the joins in the evaluation. More precisely, suppose we are given a query Q and a database $\mathcal{D}$. We assume use of only *left-linear query plans*, where selections are pushed as far as they go and projection is the last operation. Thus, each plan is a permutation of the subgoals of the query, and the cost of this query plan on a given database instance $\mathcal{D}$ is defined inductively as follows. For $n = 1$, the cost of query plan $Q = R_1$ is the size of the relation R_1. For each $n \geq 2$, the cost of query plan $(\ldots((R_1 \bowtie R_2) \bowtie R_3) \bowtie \ldots \bowtie R_n)$ over n relations is the sum of the following four values:

1. the cost of query plan $(\ldots((R_1 \bowtie R_2) \bowtie R_3) \bowtie \ldots \bowtie R_{n-1})$
2. the size of relation $R_1 \bowtie \ldots \bowtie R_{n-1}$
3. the size of relation R_n and
4. the size of relation $R_1 \bowtie \ldots \bowtie R_n$

The cost of evaluating a query Q on a database $\mathcal{D}$, denoted as $C(Q, \mathcal{D})$, is the minimum cost over all Q's query plans when evaluated on $\mathcal{D}$. Moreover, the cost of a query workload, denoted as $C(\mathcal{Q}, \mathcal{D})$, is defined as the sum of the costs of all queries in the workload. In this paper, although we use the above cost model, our results also hold for cost-models for which the evaluation cost is increasing with the size of intermediate relations [11,8,2,4].

4 The Space of Optimal Solutions

In this section, we elaborate on the search space analysis of candidate solutions for bag-oriented view selection problems, considering that both queries and views are conjunctive queries/views. The main results of this section are as follows: In Subsection 4.1, we propose techniques to reduce the search space of candidate views and demonstrate that if there exists a solution for a given problem input, then there is at least one optimal solution of a specific form. We refer to these solutions as the *representative (optimal) set of solutions*. In Subsection 4.2, an algorithm is presented that computes the representative set of optimal solutions.

4.1 Representative Set of Solutions

In [2], it has been proved that for workloads of conjunctive queries each view in any admissible viewset (and thus in any optimal viewset) can be defined as a generalization of a subexpression of some query in the workload. The following lemma, which combines Lemmas 2 and 3 of [2], presents this result formally:

Lemma 1. *Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be a conjunctive bag-oriented problem input, $\mathcal{V}$ be any* admissible *viewset for $\mathcal{P}$, and Q be any query in $\mathcal{Q}$. Suppose that $\mathcal{V}' \subseteq \mathcal{V}$ is the set of all views used in an equivalent rewriting R of Q in terms of $\mathcal{V}$. Then:*

1. *The subgoals in the expansion of R corresponding to the definitions of views $\mathcal{V}'$ form a partition of the (subgoals in the) definition of Q.*
2. *Each view in $\mathcal{V}'$ can be defined as a generalization of a subexpression of Q which is a member of the partition as defined in (1).*

Lemma 1 precisely describes a search space (consisting of all query subexpressions and their generalizations) to look for view definitions. As, in general, this search space is huge, it is crucial to investigate ways to reduce this search space (possibly for special cases of the view selection problem) in order to construct efficient algorithms for solving the view selection problem. A significant improvement in this direction might be to restrict the search space to contain only the subexpressions of the queries in the query workload (i.e. to exclude the generalizations of the subexpressions). Unfortunately, as it is shown in the following example, in the general case this is not possible.

Example 1. Consider a database schema $\mathcal{S}$ that contains only the relation e of arity 4 and a query workload $\mathcal{Q} = \{Q_1, Q_2\}$ on $\mathcal{S}$, where:

$$Q_1 : \ q_1(X,Y) \text{ :- } e(X,X,X,Y).$$
$$Q_2 : \ q_2(X,Y) \text{ :- } e(X,Y,Y,Y).$$

Consider also the following three viewsets $\mathcal{V}_1$, $\mathcal{V}_2$ and $\mathcal{V}_3$:

- $\mathcal{V}_1 = \{V_{11}, V_{12}\}$, where:
$$V_{11} : v_{11}(X_1,X_2) \text{ :- } e(X_1,X_1,X_1,X_2).$$
$$V_{12} : v_{12}(X_1,X_2) \text{ :- } e(X_1,X_2,X_2,X_2).$$
- $\mathcal{V}_2 = \{V_2\}$, where:
$$V_2 : v_2(X_1,X_2,X_3) \text{ :- } e(X_1,X_2,X_2,X_3).$$
- $\mathcal{V}_3 = \{V_3\}$, where:
$$V_3 : v_3(X_1,X_2,X_3,X_4) \text{ :- } e(X_1,X_2,X_3,X_4).$$

Notice that the bodies of the view definitions of $\mathcal{V}_1$ are subexpressions of the bodies of the queries in $\mathcal{Q}$ (in fact they are obtained from the bodies of Q_1 and Q_2 by renaming their variables), while the bodies of the views in $\mathcal{V}_2$ and $\mathcal{V}_3$ are generalizations of these subexpressions. Using each one of the above viewsets we get equivalent rewritings for the queries in $\mathcal{Q}$. More specifically, using $\mathcal{V}_1$ we get:

$$R_1 : r_1(X,Y) \text{ :- } v_{11}(X,Y).$$
$$R_2 : r_2(X,Y) \text{ :- } v_{12}(X,Y).$$

where R_1 and R_2 are equivalent rewritings of Q_1 and Q_2 respectively. Using $\mathcal{V}_2$ we get:

$$R'_1 : r'_1(X,Y) \text{ :- } v_2(X,X,Y).$$
$$R'_2 : r'_2(X,Y) \text{ :- } v_2(X,Y,Y).$$

where R'_1 and R'_2 are equivalent rewritings of Q_1 and Q_2 respectively. Finally, using $\mathcal{V}_3$ we get:

$$R''_1 : r''_1(X,Y) \text{ :- } v_3(X,X,X,Y).$$
$$R''_2 : r''_2(X,Y) \text{ :- } v_3(X,Y,Y,Y).$$

where R''_1 and R''_2 are equivalent rewritings of Q_1 and Q_2 respectively.

Assuming a database instance D={(e(a, a, a, a);1), (e(a, b, c, d);5)}, the sets $\mathcal{V}_1(D)$, $\mathcal{V}_2(D)$ and $\mathcal{V}_3(D)$ are:

$$\mathcal{V}_1(D) = \{(v_{11}(a,a);1), (v_{12}(a,a);1)\}.$$
$$\mathcal{V}_2(D) = \{(v_2(a,a,a);1)\}.$$
$$\mathcal{V}_3(D) = \{(v_3(a,a,a,a);1), (v_3(a,b,c,d);5)\}.$$

Since $size(\mathcal{V}_3(D))=6$, $size(\mathcal{V}_1(D))=2$ and $size(\mathcal{V}_2(D))=1$, we have $size(\mathcal{V}_3(D)) > size(\mathcal{V}_1(D)) > size(\mathcal{V}_2(D))$. If we choose a storage limit $L = size(\mathcal{V}_2(D)) = 1$, then $\mathcal{V}_2$ is the only admissible viewset among the above three.

Example 1 shows that, in some cases, any optimal solution requires views that cannot be constructed as subexpressions of the queries in the query workload.

The optimal solution in Example 1 uses views constructed using generalizations of subexpressions of the queries. In particular, the view in the optimal viewset $\mathcal{V}_2$ is defined as a common generalization of the bodies of both queries in the query workload $\mathcal{Q}$. Based on these observations two questions arise:

1. Are there any special cases of the view selection problem for which there are optimal solutions whose viewset can be constructed by considering only subexpressions of the queries in the query workload?
2. For the general case, can we reduce the search space specified by Lemma 1 which consists of all possible generalizations of query subexpressions?

Both questions can be answered affirmatively as shown in the following Propositions 1 and 2.

Proposition 1. *Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be a conjunctive bag-oriented view selection problem input such that every relation in $\mathcal{S}$ appears at most once in a body of some query in $\mathcal{Q}$. If there exists a solution for $\mathcal{P}$, then there exists an optimal solution $\Lambda = (\mathcal{V}, \mathcal{R})$ such that each view in $\mathcal{V}$ is defined as a subexpression of a query in $\mathcal{Q}$.*

Notice that, when the assumptions of Proposition 1 hold, the queries in the workload $\mathcal{Q}$ do not contain self-joins. In this case, because of Theorem 5 of [2], we can rewrite each query in $\mathcal{Q}$ without using self-joins of views in $\mathcal{V}$.

We now focus on the general case and prove, in Proposition 2, that, in order to construct an optimal viewset, we need to consider both subexpressions of queries and lgg's of subexpressions. We can thus exclude all those generalizations of subexpressions that are not lgg's of two or more subexpressions.

Proposition 2. *Let $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ be a conjunctive bag-oriented view selection problem. If there exists a solution for $\mathcal{P}$, then there is an optimal solution $\Lambda = (\mathcal{V}, \mathcal{R})$ for $\mathcal{P}$ such that the body of each view in $\mathcal{V}$ is either a subexpression of a query in $\mathcal{Q}$ or an lgg of two or more subexpressions of queries in $\mathcal{Q}$.*

The intuition behind Propositions 1 and 2 is that the use of generalization of subexpressions in defining a view is useful only when this view definition will be subsequently used two or more times to construct equivalent rewritings for the queries in the workload $\mathcal{Q}$. This is the case of the viewsets $\mathcal{V}_2$ and $\mathcal{V}_3$ in Example 1. Besides, it is not useful to generalize the subexpression more than needed as this, in general, increases the number of the tuples obtained when materializing this "overgeneralized" view definition and this does not contribute towards an improvement of the evaluation of the rewriting . An example of such "overgeneralization" is the viewset $\mathcal{V}_3$ in Example 1.

We further refine Propositions 1 and 2 by restricting also the vector of variables in the heads of the view definitions. The simplest choice is to put as arguments of a view head all different variables appearing in the view's body. However, this is not always the "best" choice as the following example shows:

Example 2. Consider a query workload $\mathcal{Q} = \{Q\}$, where:

$$Q: \ q_1(X, Y) \text{ :- } e(X, Z), f(Z, W), g(W, Y).$$

Consider also the following viewset $\mathcal{V}_1 = \{V_{11}, V_{12}\}$:

$$V_{11} : v_{11}(X, Z, W) \text{ :- } e(X, Z), f(Z, W).$$
$$V_{12} : v_{12}(W, Y) \text{ :- } g(W, Y).$$

Notice that using $\mathcal{V}_1$ as we can get the following equivalent rewriting R of Q:

$$R : r(X, Y) \text{ :- } v_{11}(X, Z, W), v_{12}(W, Y).$$

It is easy to see, however, that the variable Z in the head of V_{11} is redundant. More specifically, if we replace the view V_{11} in $\mathcal{V}_1$ by the following view V'_{11}:

$$V'_{11} : v'_{11}(X, W) \text{ :- } e(X, Z), f(Z, W).$$

we get R' which is also an equivalent rewriting of Q:

$$R' : r'(X, Y) \text{ :- } v'_{11}(X, W), v_{12}(W, Y).$$

Comparing V_{11} and V'_{11}, it is easy to see that, under bag semantics, for every database D we have $size(\mathcal{V}_{11}(D)) = size(\mathcal{V}'_{11}(D))$. Also, the query R', obtained by using V'_{11} to rewrite Q, is computed more efficiently than the rewriting R obtained by using V_{11} to rewrite Q.

We now show how to choose the appropriate set of variables to be used as head arguments of the view definitions.

Definition 4. *Let Q be a query of the form H :- $B_1, \ldots, B_n$ and $S = B_{11}, \ldots, B_{1k}$, with $1 \leq k \leq n$, be a subexpression of the body of Q. Let $Q' = Q - S$ be the query obtained by removing from the body of Q the atoms in S. Then, the set $lvars(Q, S) = Vars(Q') \cap Vars(S)$, is called the* linking variables *of Q and S.*

Example 3. (Continued from Example 2) Consider the query Q in Example 2 and the subexpression $S = e(X, Z), f(Z, W)$ of Q. It is easy to see that the set of linking variables of Q and S is $lvars(Q, S) = \{X, W\}$.

Proposition 3. *Let Q be a conjunctive query and V be a view whose body is defined as a subexpression of Q. Then the view V can be used in an equivalent rewriting of Q, if and only if $lvars(Q, S) \subseteq vars(head(V))$.*

The linking variables are related to the shared-variables property introduced by [17]; that holds in the set-oriented context.

What the above proposition indicates is that the set of linking variables is the minimum set of variables that should be put in the head of the view definition so as this view can be used in an equivalent rewriting of the query.

Example 4. (Continued from Example 3) Notice that the variables in $\{X, W\}$, which are the linking variables of Q and S, appear in the heads orf both views V_{11} and V'_{11} constructed from the subexpression S of Q. Observe that, if we

remove X or W or both from the head of the view V_{11} (or the view V'_{11}), then the corresponding viewset cannot give equivalent rewriting for the query Q.

Proposition 3 refers to views which are defined as subexpressions of the queries in the query workload. We now investigate the problem of selecting the head arguments of the views defined as least general generalizations of subexpressions of queries. For this we need the following definition:

Definition 5. *Let $E = \{S_1, \ldots, S_k\}$, with $k > 1$, be a set of expressions, and G be their least general generalization (supposing that such an lgg exists). Let S be an expression in E and M be a mapping for the arguments of S to the arguments of G such that each argument of S in a position (i, j), where i is the order of an atom in S and j is the order of the argument in the i-th atom of S, maps to the argument which is in the position (i, j) on G. Let X be a variable in $vars(S)$. Then the* corresponding variable set *of X in G is defined as $\{Y | X$ appears in a position (i, j) of S and Y is the variable in the position (i, j) of $G\}$.*

Proposition 4 specifies the minimum set of variables that should be put in the head of a view defined as the lgg of two or more subexpressions.

Proposition 4. *Let $Q_1, \ldots, Q_k$, with $k > 1$, be (not necessarily different) queries in a query workload $\mathcal{Q}$, and let $S_1, \ldots, S_k$, be expressions such that S_i is a subexpression of Q_i for $1 < i \leq k$. Suppose that the least general generalization of $S_1, \ldots, S_k$ exists and that V is a view whose body is the least general generalization of $S_1, \ldots, S_k$. Then the view V can be used in an equivalent rewriting of Q_i, for all $i = 1, \ldots, k$, if and only if $\bigcup_{i=1}^{k} L_i \cup \bigcup_{i=1}^{k} M_i \subseteq vars(head(V))$, where (a) L_i is the union of the corresponding variable sets of the variables in $lvars(Q_i, S_i)$, and (b) M_i is the union of the corresponding variable sets of the variables in S_i whose corresponding variable sets are not singletons.*

Another way to construct the view V whose body is the least general generalization of the subexpressions S_1 and S_2 of two queries Q_1 and Q_2 respectively proceeds in two steps as follows:

1. We construct the views V_1 and V_2 using the subexpressions S_1 and S_2 respectively as bodies and the linking variables with Q_1 and Q_2 as head variables.
2. By considering V_1 and V_2 as queries we construct V with body the lgg of the bodies of V_1 and V_2 and with head variables the minimum set of variables specified by Proposition 4. V is said to be an *lgview* of the views V_1 and V_2.

This procedure can be easily generalized for more than two subexpressions.

An interesting question referring to lgviews is the following: "Does the inequality $size(V) \leq size(V_1) + size(V_2)$ always hold for the lgview V of two views V_1 and V_2?". Notice that, if the answer is "yes" for any bag-oriented view selection problem input, then whenever an lgview exists, the original views can be discarded eliminating in this way the search space for finding viewsets. Unfortunately, the inequality does not always hold, as the following example shows.

Example 5. Let a viewset $\mathcal{V} = \{V_1, V_2\}$, where the definitions of the views are:

$$V_1 : v_1(X, Z) \text{ :- } p_1(X, X), p_2(X, Z).$$
$$V_2 : v_2(X, Z) \text{ :- } p_1(X, Z), p_2(Z, Z).$$

where p_1 and p_2 are binary relations on the database schema $\mathcal{S}$. Consider also another viewset $\mathcal{W} = \{W\}$ whose view W is defined as:

$$W : w(A, B, C) \text{ :- } p_1(A, B), p_2(B, C).$$

Notice that W is the lgview of the views in $\mathcal{V}$. Assuming the database instance:

$$\mathcal{D} = \{p_1(1,1), p_1(1,2), p_1(3,4), p_2(1,1), p_2(1,2), p_2(2,2), p_2(2,3), p_2(4,5)\},$$

in which the multiplicity of each database tuple in this example is 1 and for this we omit it, and materializing the views over this database we get:

$$\mathcal{V}(\mathcal{D}) = \{v_1(1,1), v_1(1,2), v_2(1,1), v_2(1,2)\}.$$
$$\mathcal{W}(\mathcal{D}) = \{w(1,1,1), w(1,1,2), w(1,2,2), w(1,2,3), w(3,4,5)\}.$$

It is easy to see that $\text{size}(\mathcal{V}(\mathcal{D})) < \text{size}(\mathcal{W}(\mathcal{D}))$.

The following theorem summarizes the results of this section:

Theorem 1. *Let a bag-oriented view selection input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$. If there is a solution for $\mathcal{P}$, then there exists an optimal solution $\Lambda = (\mathcal{V}, \mathcal{R})$ such that each view in $\mathcal{V}$ is either a subexpression view or an lgview whose body is constructed as specified by Proposition 2, and whose head is constructed using the minimal set of variables specified by Propositions 3 and 4, respectively.*

Thus the class of solutions constructed as above is a *representative set of solutions* for a given bag-oriented view selection problem input $\mathcal{P}$.

4.2 LGG-VSB Algorithm

An algorithm, called LGG-VSB, which is based on the results of the previous section, and outputs the representative set of optimal solutions, for a given view selection problem input, is proposed in this section. LGG-VSB incorporates the results of the Theorem 1 and Lemma 1 to the algorithm CGALG (introduced in [2]), reducing significantly the search space for finding an optimal solution. In particular, LGG-VSB avoids the construction of viewsets that do not rewrite the queries in the workload, by producing the candidate viewsets in such a way that the construction of the equivalent rewritings of the query is quickly achieved; i.e. instead of construction of every set of views whose body is a generalization of a subexpression of a query's body (CGALG), LGG-VSB constructs viewsets that form a partition of the body of each query in the workload.

Algorithm *LGG-VSB*.
Input: A bag oriented view selection problem input[1] $\mathcal{P} = \{\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L}\}$.
Output: Λ, the representative set of optimal solutions.
Begin
1. Let $\mathcal{V}$ be a set of viewsets constructed as follows: Each $\mathcal{V}' \in \mathcal{V}$ is of the form $\mathcal{V}' = \mathcal{V}_1 \cup \ldots \cup \mathcal{V}_n$, where n is the number of queries in $\mathcal{Q}$ and each viewset $\mathcal{V}_i$ is obtained from the query $Q_i \in \mathcal{Q}$ as follows:
   - Let P_i be a partition of the subgoals of Q_i.
   - For each block $B_j \in P_i$, add a view definition $V_{i,j}$ in $\mathcal{V}_i$ whose body consists of the atoms in B_j and whose head variables are the variables in $lvars(Q_i, B_j)$.
2. Set $G_0 = \mathcal{V}$; set $i = 0$.
3. **while** $G_i \neq \emptyset$ **do**
   - $G_{i+1} = \{\mathcal{V}_g | \mathcal{V}_g = (\mathcal{V}' - \mathcal{M}) \cup \{V_l\}$, where $\mathcal{V}' \in G_i$ and $\mathcal{M} \subseteq \mathcal{V}'$ and $V_l = lgview(\mathcal{M})\}$.
   - i = i + 1.

   end while
4. Let $\mathcal{V} = \bigcup_{j=0,\ldots,i} G_j$.
5. Compute the cost $C(\mathcal{Q}, \mathcal{D})$ of $\mathcal{Q}$ on $\mathcal{D}$ and set it to C_{opt}.
6. **For** every viewset $\mathcal{V}' \in \mathcal{V}$, such that $size(\mathcal{V}') \leq L$, **do**
   - Construct the set $\mathcal{R}_{\mathcal{V}'}$ of all equivalent rewritings of $\mathcal{Q}$ using $\mathcal{V}'$.
   - Set $\Lambda = \emptyset$.
   - **For** every distinct subset $\mathcal{R}$ of $\mathcal{R}_{\mathcal{V}'}$ such that $\mathcal{R}$ contains an equivalent rewriting of each query in $\mathcal{Q}$, **do**
     - Let $c = C(\mathcal{R}, \mathcal{V}'(\mathcal{D}))$.
     - **If** $c < C_{opt}$, **then** set $C_{opt} = c$ and set $\Lambda = \{(\mathcal{V}', \mathcal{R})\}$ **else if** $c = C_{opt}$, **then** $\Lambda = \Lambda \cup \{(\mathcal{V}', \mathcal{R})\}$.

end.

5 Chain and Path Queries

In this section, we study the bag-oriented view selection problem when the query workload is a set of either chain queries or path queries. The main results are as follows: Subsection 5.1 demonstrates that for a problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, where $\mathcal{Q}$ is a workload of chain queries, we cannot restrict the space of optimal solutions by searching admissible viewsets which contain only *chain-views*, i.e. views defined by chain queries. Subsection 5.2 demonstrates that for a problem input $\mathcal{P} = (\mathcal{S}, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, where $\mathcal{Q}$ is a workload of path queries, if there exists a solution for $\mathcal{P}$, then there is at least one optimal solution for $\mathcal{P}$ which is constructed by an admissible viewset containing only path views (Theorem 2).

5.1 Chain-Query Workload

In this section we study the view selection problem for workloads containing only chain-queries. In particular, we focus our attention on whether there is an

[1] Recall that $\mathcal{L} = \{L\}$, where L is a single storage limit constraint.

optimal solution constructed by a set of chain-views. Unfortunately, as the following proposition shows, there are cases in which none of the optimal solutions is constructed by a set of chain-views.

Proposition 5. *There exists at least one bag-oriented view selection problem input $\mathcal{P} = (S, \mathcal{Q}, \mathcal{D}, \mathcal{L})$ such that:*

- *$\mathcal{Q}$ is a set of chain queries, and*
- *$\mathcal{P}$ has optimal solutions but there is **no** optimal solution $\Lambda = (\mathcal{V}, R)$ such that $\mathcal{V}$ contains only chain queries.*

Proof. The following example proves this proposition.

Example 6. Consider a query workload $\mathcal{Q} = \{Q\}$ on a database schema $\mathcal{S}$ that contains the binary relations r_1, r_2 and r_3, where Q is the following chain query:

$$Q : q(X, Y) \text{ :- } r_1(X, Z), r_2(Z, W), r_3(W, Y).$$

Consider also the following five viewsets $\mathcal{V}_i$, $i \in \{1, 2, 3, 4, 5\}$:

$\mathcal{V}_1 = \{V_{11}, V_{12}\}$, where:
$$V_{11} : v_{11}(X, Z, W, Y) \text{ :- } r_1(X, Z), r_3(W, Y).$$
$$V_{12} : v_{12}(X, Y) \text{ :- } r_2(X, Y).$$
$\mathcal{V}_2 = \{V_{21}, V_{22}\}$, where:
$$V_{21} : v_{21}(X, Y) \text{ :- } r_1(X, Z), r_2(Z, Y).$$
$$V_{22} : v_{22}(X, Y) \text{ :- } r_3(X, Y).$$
$\mathcal{V}_3 = \{V_{31}, V_{32}\}$, where:
$$V_{31} : v_{31}(X, Y) \text{ :- } r_2(X, Z), r_3(Z, Y).$$
$$V_{32} : v_{32}(X, Y) \text{ :- } r_1(X, Y).$$
$\mathcal{V}_4 = \{V_{41}\}$, where:
$$V_{41} : v_{41}(X, Y) \text{ :- } r_1(X, Z), r_2(Z, W), r_3(W, Y).$$
$\mathcal{V}_5 = \{V_{51}, V_{52}, V_{53}\}$, where:
$$V_{51} : v_{51}(X, Y) \text{ :- } r_1(X, Y).$$
$$V_{52} : v_{52}(X, Y) \text{ :- } r_2(X, Y).$$
$$V_{53} : v_{53}(X, Y) \text{ :- } r_3(X, Y).$$

Observe that the above viewsets are all possible viewsets constructed as described in Section 4.

Suppose that we are given database instance $\mathcal{D} = \{(r_1(\text{a,b});5), (r_2(\text{b,c});10), (r_3(\text{c,d});5)\}$. Considering a storage limit L=35 tuples, the following viewsets:

$$\mathcal{V}_1(\mathcal{D}) = \{(v_{11}(a, b, c, d); 25), (v_{12}(b, c); 10)\}$$
$$\mathcal{V}_5(\mathcal{D}) = \{(v_{51}(a, b); 5), (v_{52}(b, c); 10), (v_{53}(c, d); 5)\}$$

do not violate the storage limit constraint. In contrast, the viewsets:

$$\mathcal{V}_2(\mathcal{D}) = \{(v_{21}(a, c); 50), (v_{22}(c, d); 5)\}$$
$$\mathcal{V}_3(\mathcal{D}) = \{(v_{31}(a, c); 50), (v_{32}(c, d); 5)\}$$
$$\mathcal{V}_4(\mathcal{D}) = \{(v_{41}(a, c); 250)\}$$

do violate it. Thus, $\Lambda = (\mathcal{V}_1, R)$ and $\Lambda' = (\mathcal{V}_5, R')$ are solutions for input $\mathcal{P}$, where the rewritings R and R' are the following:

$$R : q(X, Y) \text{ :- } v_{11}(X, Z, W, Y), v_{12}(Z, W).$$
$$R' : q(X, Y) \text{ :- } v_{51}(X, Z), v_{52}(Z, W), v_{53}(W, Y).$$

Using the cost model presented in Section 3, the costs of Λ and Λ' are $C(R, \mathcal{V}_1(\mathcal{D})) = 55$ and $C(R', \mathcal{V}_4(\mathcal{D})) = 325$ respectively. As a consequence, Λ is an optimal solution for $\mathcal{P}$.

5.2 Path-Query Workload

In this section we study the view selection problem for *path-query workloads* (i.e. workloads of path queries). Unlike to the problem for chain query workloads in which we cannot reduce the search space to the class of chain views, for path-query workloads we can reduce the search space even more. The main result of this section, presented by the following theorem, is that whenever the workload is a set of path-queries, we can focus on *path-viewsets* whose views have at most as many subgoals as the length of the longest path-query in the workload.

Theorem 2. *Let $\mathcal{P} = (S, \mathcal{Q}, \mathcal{D}, \mathcal{L})$, be a conjunctive bag-oriented view selection input, and $\mathcal{Q}$ contains a set of path queries. If there exists a solution $\Lambda = (\mathcal{V}_o, \mathcal{R}_o)$ for $\mathcal{P}$, then there is an optimal solution $\Lambda' = (\mathcal{V}'_o, \mathcal{R}'_o)$ for $\mathcal{P}$ such that:*

- *each view in $\mathcal{V}'_o$ is defined as a path of the same relation as a query $Q \in \mathcal{Q}$,*
- *every view in $\mathcal{V}'_o$ has at most n subgoals, where n is the length of the longest query in $\mathcal{Q}$,*
- *every $R \in \mathcal{R}'_o$ is a chain query.*

Consequently, we may restrict our attention in searching optimal solutions constructed by path-viewsets. In this case, the number of admissible viewsets is exponential to the number of subgoals of the path-queries in the workload. This exponential bound is implied by the reduction of the problem of searching path-viewsets to the integer-partitioning problem [5].

Based on Theorem 2, we can improve the LGG-VSB for workloads containing only path-queries.In particular, when we know that the workload $\mathcal{Q}$ consists of n path-queries of the same relation, steps 1-4 of LGG-VSB can be replaced by:

- Each $\mathcal{V}_\mathcal{I} \in \mathcal{V}$ contains a path-view V_k of length k, for every distinct integer $k \in \mathcal{I}$, where the set of integers $\mathcal{I}$ is of the form $\mathcal{I} = \mathcal{I}_{k_1} \cup \ldots \cup \mathcal{I}_{k_n}$, and $\mathcal{I}_{k_i}$ is a partition of the length of path-query $P_{k_i} \in \mathcal{Q}$, $i \in \{1, \ldots n\}$; the partitions of an integer can be computed using an algorithm from [26].

6 Conclusion

In this paper we studied the problem of view selection under bag semantics. In particular, we investigated ways to limit the search space of candidate views,

given a workload of CQs. We improved previous results by exploiting very refined characterizations of views that participate in equivalent rewritings. Based on these characterizations we proposed sound and complete algorithms to select views for a query workload. Besides, we studied the problem in two special cases, that is, when the workload contains only (a) chain queries, or (b) path queries, and present interesting results which further improve the proposed algorithm. Concerning the experimental evaluation of our approach, we have contacted preliminary experiments that gave promising results.

There is a lot to be done for future work including the following: (a) studying further the potential features of lgviews, (b) studying more special cases of the view selection problem, (c) studying the view selection problem for parameterized queries, and (d) studying the exact complexity of the problem.

Acknowledgements. We would like to thank Timos Sellis and the anonymous reviewers for their valuable comments.

References

1. Abiteboul, S., Hull, R., Vianu, V.: Foundations of Databases. Addison-Wesley, Reading (1995)
2. Afrati, F., Chirkova, R., Gergatsoulis, M., Pavlaki, V.: View selection for real conjunctive queries. Acta Inf. 44(5), 289–321 (2007)
3. Afrati, F.N., Chirkova, R.: Selecting and using views to compute aggregate queries (extended abstract). In: Eiter, T., Libkin, L. (eds.) ICDT 2005. LNCS, vol. 3363, pp. 383–397. Springer, Heidelberg (2004)
4. Afrati, F.N., Li, C., Ullman, J.D.: Generating efficient plans for queries using views. In: SIGMOD Conference 2001, pp. 319–330 (2001)
5. Andrews, G.E., Eriksson, K.: Integer Partitions. Cambridge University Press, Cambridge (2004)
6. Baralis, E., Paraboschi, S., Teniente, E.: Materialized views selection in a multidimensional database. In: VLDB 1997, pp. 156–165 (1997)
7. Chirkova, R., Genesereth, M.R.: Linearly bounded reformulations of conjunctive databases. In: Palamidessi, C., Moniz Pereira, L., Lloyd, J.W., Dahl, V., Furbach, U., Kerber, M., Lau, K.-K., Sagiv, Y., Stuckey, P.J. (eds.) CL 2000. LNCS, vol. 1861, pp. 987–1001. Springer, Heidelberg (2000)
8. Chirkova, R., Halevy, A.Y., Suciu, D.: A formal perspective on the view selection problem. The VLDB Journal 11(3), 216–237 (2002)
9. Chirkova, R., Li, C.: Materializing views with minimal size to answer queries. In: PODS, pp. 38–48 (2003)
10. Florescu, D., Levy, A.Y., Suciu, D., Yagoub, K.: Optimization of run-time management of data intensive web-sites. In: VLDB 1999, pp. 627–638 (1999)
11. Gupta, H., Harinarayan, V., Rajaraman, A., Ullman, J.D.: Index selection for OLAP. In: ICDE 1997, pp. 208–219 (1997)
12. Gupta, H., Mumick, I.S.: Selection of views to materialize in a data warehouse. IEEE Trans. Knowl. Data Eng. 17(1), 24–43 (2005)
13. Harinarayan, V., Rajaraman, A., Ullman, J.D.: Implementing data cubes efficiently. SIGMOD Rec. 25(2), 205–216 (1996)
14. Karloff, H., Mihail, M.: On the complexity of the view-selection problem. In: PODS 1999, pp. 167–173 (1999)

15. Lloyd, J.W.: Foundations of logic programming. Springer, Heidelberg (1984)
16. Plotkin, G.: A note on inductive generalization. Machine Intelligence 5, 153–163 (1970)
17. Pottinger, R., Halevy, A.: Minicon: A scalable algorithm for answering queries using views. The VLDB Journal 10(2-3), 182–198 (2001)
18. Rizzi, S., Saltarelli, E.: View materialization vs. indexing: Balancing space constraints in data warehouse design. In: Eder, J., Missikoff, M. (eds.) CAiSE 2003. LNCS, vol. 2681, pp. 502–519. Springer, Heidelberg (2003)
19. Surajit Chaudhuri, M., Vardi, M.Y.: Optimization of real conjunctive queries. In: PODS 1993, pp. 59–70 (1993)
20. Theodoratos, D., Sellis, T.K.: Data warehouse configuration. In: VLDB 1997, pp. 126–135 (1997)
21. Theodoratos, D., Xu, W.: Constructing search spaces for materialized view selection. In: DOLAP, pp. 112–121 (2004)
22. Ullman, J.D., Garcia-Molina, H., Widom, J.: Database Systems: The Complete Book. Prentice Hall PTR, Upper Saddle River (2001)
23. Xu, W., Theodoratos, D., Zuzarte, C.: Computing closest common subexpressions for view selection problems. In: DOLAP, pp. 75–82 (2006)
24. Yu, J.X., Choi, C.-H., Gou, G., Lu, H.: Selecting views with maintenance cost constraints: Issues, heuristics and performance. Journal of Research and Practice in Information Technology 36(2), 89–110 (2004)
25. Zhou, J., Larson, P.-A., Freytag, J.C., Lehner, W.: Efficient exploitation of similar subexpressions for query processing. In: SIGMOD Conference, pp. 533–544 (2007)
26. Zoghbi, A., Stojmenović, I.: Fast algorithms for generating integer partitions. Int. J. Comput. Math. 70(2), 319–332 (1998)