Constructing Search Spaces for Materialized View Selection

Dimitri Theodoratos
Department of Computer Science
New Jersey Institute of Technology
dth@cs.njit.edu

Wugang Xu
Department of Computer Science
New Jersey Institute of Technology
wx2@njit.edu

ABSTRACT

Deciding which views to materialize is an important problem in the design of a Data Warehouse. Solving this problem requires generating a space of candidate view sets from which an optimal or near-optimal one is chosen for materialization. In this paper we address the problem of constructing this search space. This is an intricate issue because it requires detecting and exploiting common subexpressions among queries and views. Our approach suggests adding to the alternative evaluation plans of multiple queries views called closest common derivators (CCDs) and rewriting the queries using CCDs. A CCD of two queries is a view that is as close to the queries as possible and that allows both queries to be (partially or completely) rewritten using itself. CCDs generalize previous definitions of common subexpressions. Using a declarative query graph representation for queries we provide necessary and sufficient conditions for a view to be a CCD of two queries. We exploit these results to describe a procedure for generating all the CCDs of two queries and for rewriting the queries using each of their CCDs.

Categories and Subject Descriptors

H.2 [**DATABASE MANAGEMENT**]: Logical Design; H.2.7 [**DATABASE MANAGEMENT**]: Database Administration—*Data warehouse and repository*

General Terms

Data warehouse and repository design

Keywords

query, materialized view, query rewriting, query graph, common subexpression

DOLAP'04, November 12–13, 2004, Washington, DC, USA.

1. INTRODUCTION

One of the most important issues in Data Warehousing is the design of a Data Warehouse. This issue can be abstractly modeled as a problem (called materialized view selection problem) that takes as input a set of queries and a number of cost-determining parameters (e.g. query frequencies or source relation change propagation frequencies). The output is a set of views to materialize in the Data Warehouse that minimizes a cost function (e.g. query evaluation cost, view maintenance cost, or their combination) and satisfies a number of constraints (e.g. storage space restrictions or view maintenance cost constraints) [23]. Solving the materialized view selection problem involves addressing two main tasks: (1) generating a search space of alternative view sets for materialization, and (2) designing optimization algorithms that select an optimal or near-optimal view set from the search space. There is a substantial body of work dealing with the second task, often suggesting elaborate greedy, heuristic or randomized optimization algorithms [13, 2, 28, 11, 22, 12, 17, 26, 18, 15]. However, these approaches assume that the search space is available, usually in the form of multiple common evaluation plans for all the input queries represented as an AND/OR graph (for the case of general queries) or in the form of multidimensional lattices of views (for the case of star grouping/aggregation queries). Even though the construction of multidimensional lattices is straightforward [13, 2, 22], the construction of alternative common plans for multiple general queries is an intricate issue. The reason is that common subexpresions [14] among the input queries need to be detected and exploited, and the queries need to be rewritten using these common subexpressions. In this paper we deal with the problem of constructing a search space for materialized view selection. Clearly, this is an important problem since, if the search space does not include a view selection that is close to the optimal, the optimization algorithms will fail to produce satisfactory results.

1.1 Previous work.

The need to construct evaluation plans for multiple queries by exploiting common subexpressions first appeared in the area of multiquery optimization [21]. The goal there is to determine a global evaluation plan (a plan for all the queries together) that is more efficient to evaluate than evaluating separately the optimal evaluation plan of each query. Initially "common subexpression" referred to identical or equivalent expressions [6]. Subsequently the term included subexpression subsumption [14, 3, 21]. Later, commonalities between the queries included the possibility for overlapping

selection conditions [4]. A typical approach for constructing evaluation plans for multiple queries is to proceed in a bottom-up way (from the base relations to the roots of the plans) and to combine nodes from different plans that are defined over the same set of base relations. Equivalent nodes are merged, nodes related through subsumption are linked with selection operations, and overlapping nodes are linked through new nodes that are able to compute the overlapping nodes. The disadvantage of this approach is that all the alternative plans of the input queries need to be considered in order to guarantee that the optimal view selection is included in the search space. Clearly, it is not feasible to generate and combine all the alternative plans of the input queries. This approach is followed in [1] where merge rules are suggested for plans of queries involving selection, projection, join and grouping/aggregation operations. In this work, however, node combination is restricted to view subsumption (node overlapping is ignored) and there are no results as to the completeness of the suggested rules. [8, 7] propose an algorithm to determine a set of candidate views for materialization for queries and views that are nested expressions involving selection, projection, join and grouping/aggregation operations. New views are generated in the search space by computing the ancestor of two expressions, i.e. the coarsest view using which both expressions can be completely rewritten [19, 5]. This approach is restricted to nested queries and views involving star joins over multidimensional star schemas. [25, 24] provide transformation rules for generating candidate views for materialization from a set of input select-project-join queries. The transformation rules are used by heuristic optimization algorithms that prune the space of views generated in a cost-based fashion. A restriction of this work is that self-joins are not allowed in queries and views.

1.2 Our approach and contribution.

Our approach for creating a search space for view selection consists in adding to the alternative evaluation plans of multiple queries views called closest common derivators and in rewriting the queries using these closest common derivators. A closest common derivator of two queries is a view that is as close to the queries as possible and that allows both queries to be (partially or completely) rewritten using itself. Intuitively, a closest common derivator does not involve join operations that are not necessary for computing both queries. A traditional query optimizer can be used to generate alternative evaluation plans for a closest common derivator and to choose the cheapest one among them. It can also be used to access the cost of computing each of the queries using a closest common derivator by viewing this last one as a base relation. A view selection optimization algorithm can choose to materialize a closest common derivator and/or any of the nodes (views) in its alternative evaluation plans in a cost-based manner. It is worth noting that a closest common derivator is defined using the definitions of the queries, independently of any of their evaluation plans. Further, a closest common derivator can be determined for queries that involve several occurrences of the same relation (self-joins) and are not defined over the same set of base relations.

The main contributions of the paper are the following:

- We define a common derivator of two queries as a view that can be used to rewrite both queries. Based on a closeness relationship on common derivators we formally introduce the concept of closest common derivator. This concept generalizes previous definitions of common subexpressions since the queries can be not only completely but also partially rewritten using a closest common derivator.
- We show under what conditions selection and join predicates can be merged and how their merge can be computed. The merge of two predicates is the least restrictive predicate that is implied by both predicates, and it is used in the computation of closest common derivators of queries.
- We use a declarative graph-based representation of queries and views (query graphs) and we provide necessary and sufficient conditions for a query graph to be a closest common derivator of two queries in terms of query graphs.
- We exploit the previous results to outline a procedure for computing all the closest common derivators of two queries. This procedure can be used within a view selection algorithm that can decide in a cost-based manner which closest common derivators to generate and which ones to exclude from generation, thus pruning the search space of alternative view sets for materialization.

1.3 Outline.

The rest of the paper is organized as follows. The next section provides initial definitions, determines the class of queries considered, and introduces different types of rewritings. Section 3 formally defines closest common derivators. In Section 4 we deal with the merging of query conditions. Closest common derivators in terms of query graphs and main results are presented in Section 5. We conclude in Section 6 and we suggest extensions and directions for future work. Because of lack of space proofs are omitted.

2. CLASS OF QUERIES CONSIDERED AND INITIAL DEFINITIONS

2.1 Query language.

We consider queries and views involving selections, projections, join and relation renaming operations under set semantics. Base relations can have multiple occurrences in a query. We assume that if a relation has more than one occurrence in a query, every one of its occurrences is renamed with a distinct name. The notation $R_i[R]$ denotes an occurrence of relation R renamed R_i. The *name* of a relation occurrence in a query is its new name, in case it is renamed, or its original name otherwise. Since the names of relation occurrences in a query are distinct we refer to relation occurrences in a query by their name. Attributes range over dense totally ordered domains (e.g. real or rational numbers or string spaces) [1]. Relation restrictions in queries are conditions formed as conjunctions of selection and join atomic conditions. A *join atomic condition* is an expression of the form $A\ \theta\ B + c$ where A and B are relation attributes from different relation occurrences, c is an integer or real value, and $\theta \in \{<, \leq, =, \geq, >\}$. A *selection atomic condition* is an expression of the form $A\ \theta\ B + c$ or of the from $A\ \theta\ c$

[1]However, our results equally apply to attributes ranging over discreet totally ordered domains (e.g. integers)

where A and B are relation attributes of the same relation occurrence, and c and θ are as before.

It is well known that this type of queries can be represented by relational algebra expressions of the form $\Pi_X(\sigma_C(\mathbf{R}))$ where $\mathbf{R}$ is the Cartesian product of a number of relation occurrences; σ_C denotes the selection operator with C being a condition involving attributes from the relations in $\mathbf{R}$; Π_X denotes a projection operator with X being a nonempty set of attributes from the relations in $\mathbf{R}$ (*projected attributes* of the query.)

2.2 Query containtment and equivalence.

Query Q_1 *contains* query Q_2 (denoted $(Q_1 \sqsubseteq Q_2)$ if there is a renaming of the projected attributes of Q_1 (or Q_2) that makes the answers of the two queries have the same schema and, for every instance of the base relations, the instance of the answer of Q_1 under the common schema be a subset of the instance of the answer of Q_2. Two queries Q_1 and Q_2 are *equivalent* (denoted $Q_1 \equiv Q_2$) if $Q_1 \sqsubseteq Q_2$ and $Q_2 \sqsubseteq Q_1$.

2.3 Class of queries and views.

We assume that queries and views are not equivalent to a query of the form $Q_1 \times Q_2$, where Q_1 and Q_2 are queries, and $\times$ is the Cartesian product operator. "Pure" Cartesian products are rarely used in practice and can generate huge relations. If necessary, a query $Q_1 \times Q_2$ can be represented by the two subqueries Q_1 and Q_2.

2.4 Query rewritings.

A *rewriting* of a query Q is a query equivalent to Q. A *rewriting Q' of a query Q using a view V* is a query that references V and possibly base relations such that replacing V in Q' by its definition results in a query equivalent to Q. Query rewritings using views can also involve attribute renaming operations. If there is a rewriting of Q using V we say that Q can be rewritten using V and we write $Q \vdash V$. A *complete rewriting of a query Q using a view V* is a rewriting of Q that references only V (and no base relations.) If there is a complete rewriting of Q using V we say that Q can be completely rewritten using V and we write $Q \Vdash V$. A rewriting of Q using V that is not a complete rewriting of Q using V is called *partial rewriting of Q using V*. Clearly, a partial rewriting references also base relations.

2.5 Simple rewritings.

We are interested in rewritings where each view has a single occurrence in the rewriting. To this end we introduce simple rewritings.

DEFINITION 2.1. A rewriting Q' of a query Q using a view V is *simple* if view V has a single occurrence in Q'. □

If there is a rewriting of a query Q using a view V, then there is also a simple rewriting of Q using V. This statement is not necessarily true for complete rewritings. It is possible that there is no complete simple rewriting of a query Q using a view V even if there is a complete rewriting of Q using V.

EXAMPLE 2.1. Consider the query $Q = R_1[R] \bowtie_{R_1.A<R_2.B} R_2[R]$ and the view $V = R_3[R]$. The query $Q' = V_1[V] \bowtie_{V_1.A<V_2.B} V_2[V]$ is a complete rewriting of Q using V. Clearly, there is no simple complete rewriting of Q using V. □

In the following *we consider only simple rewritings*, and by 'rewriting' we mean 'simple rewriting'. This assumption concerns both complete and partial rewritings. Based on the previous definitions we can show the following proposition.

PROPOSITION 2.1. Let Q_1 and Q_2 be two queries. If $Q_1 \Vdash Q_2$ and $Q_2 \Vdash Q_1$, $Q_1 \equiv Q_2$. □

2.6 Minimal rewritings.

We are also interested in rewritings where the computation done in the view is not repeated when evaluating the query using the view materialization. To this end we introduce minimal rewritings.

DEFINITION 2.2. A rewriting Q' of a query Q using a view V is *minimal* if for every relation R that has n, $n > 0$, occurrences in Q, R has k, $0 \leq k \leq n$, occurrences in V and $n - k$ occurrences in Q'. If there is a minimal rewriting of Q using V we say that Q can be minimally rewritten using V, and we write $Q \vdash_m V$. □

EXAMPLE 2.2. Consider the query $Q = \Pi_{R_1.A}(R_1[R] \bowtie_{R_1.A<R_2.B} R_2[R] \bowtie_{R_2.C<R_3.D} R_3[R])$ and the view $V = \Pi_{R_4.A,\, R_5.B,\, R_5.C}(R_4[R] \bowtie_{R_4.A \leq R_5.B} R_5[R])$.
$Q' = \Pi_{V.A}(V \bowtie_{V.A<R_2.B} R_2[R] \bowtie_{R_2.C<R_3.D} R_3[R])$ is a rewriting of Q using V that is not minimal: relation R has three occurrences in Q, two in V and two in Q'. In contrast, $Q'' = \Pi_{V.A}(\sigma_{V.A<V.B}(V) \bowtie_{V.C<R_3.D} R_3[R])$ is a minimal rewriting of Q using V. □

Roughly speaking, under the assumptions about redundant relation occurrences stated below, a minimal rewriting of a query Q using a view V has the minimum number of base relation occurrences among all the rewritings of Q using V. Notice that there might not exist a minimal rewriting of a query Q using a view V even if a rewriting of Q using V exists. This can happen if the attributes required for a minimal rewriting are not projected out in the view.

EXAMPLE 2.3. Consider the query $Q = \Pi_{R_1.A}(R_1[R] \bowtie_{R_1.A<R_2.B} R_2[R] \bowtie_{R_2.C<R_3.D} R_3[R])$ of Example 2.2 above and the view $V' = \Pi_{R_4.A,\, R_5.B}(R_4[R] \bowtie_{R_4.A \leq R_5.B} R_5[R])$. View V' is similar to view V of Example 2.2 with the exception of attribute $R_5.C$ which is not projected out in V'. Query $Q' = \Pi_{V'.A}(V' \bowtie_{V'.A<R_2.B} R_2[R] \bowtie_{R_2.C<R_3.D} R_3[R])$ is a rewriting of Q using V' that is not minimal. One can see that there is no minimal rewriting of Q using V' since attribute $R_5.C$ required to join V' with R_3 is not projected out in V'. □

A minimal rewriting of a query Q using a view V may not exist even if a complete rewriting of Q using V exists. In this case, the attributes required for a minimal rewriting are projected out in the view V. Therefore, what prevents the existence of a minimal rewriting is the number of occurrences of a relation in view V which may be greater than that of the same relation in query Q.

EXAMPLE 2.4. Consider the query $Q = \Pi_{R_1.A}(R_1[R])$, which has a single occurrence of R, and the view $V = \Pi_{R_2.A,\, R_3.C}(R_2[R] \bowtie_{R_2.B \leq R_3.B} R_3[R])$, which has two occurrences of R. Query $Q' = \Pi_{V.A}(V)$ is a complete rewriting of Q using V. Clearly, Q cannot be minimally rewritten using V. □

The next proposition shows that queries that can be minimally rewritten using each other are equivalent.

PROPOSITION 2.2. Let Q_1 and Q_2 be two queries. If $Q_1 \vdash_m Q_2$ and $Q_2 \vdash_m Q_1$, $Q_1 \equiv Q_2$. □

2.7 Redundant relation occurrences.

The definition of minimal rewritings above introduces some dependency on the syntax of queries and views: a minimal rewriting of a query Q using a view V might not exist even if a minimal rewriting of an equivalent query Q' using V exists. Similarly, a minimal rewriting of a query Q using a view V might not exist even if a minimal rewriting of an Q using a view V' equivalent to V exists. The reason is that query Q and/or view V may contain redundant relation occurrences. We refute this dependency below after formally defining redundant relation occurrences.

DEFINITION 2.3. A relation R has a *redundant* occurrence in a query Q, if Q can be rewritten with fewer occurrences of R. □

EXAMPLE 2.5. Consider the query $Q = \Pi_{R_1.A}(R_1[R] \bowtie_{R_1.B \leq R_2.B} R_2[R] \bowtie_{R_2.C \leq R_3.C} R_3[R])$. Query Q has two redundant occurrences of relation R: the query $Q' = \Pi_{R_4.A}(R_4[R])$, which has a single occurrence of R, is equivalent to Q. Consider also the view $V = \Pi_{R_5.A,\, R_6.C}(R_5[R] \bowtie_{R_5.B \leq R_6.B} R_6[R])$. It is not difficult to see that view V does not have any redundant relation occurrences. Query $Q'' = \Pi_{V.A}(V \bowtie_{V.C=R_3.C} R_3[R])$ is a minimal rewriting of Q using V. Clearly, Q' cannot be minimally rewritten using V. □

Detecting redundant relation occurrences and minimizing queries, that is, removing redundant relation occurrences from queries, has been studied in the past for the class of queries considered here [16]. In the following we assume that queries and views do not contain redundant relation occurrences. This assumption guarantees that minimal rewritings are independent of the syntax of queries and views.

3. CLOSEST COMMON DERIVATORS OF QUERIES

We inroduce in this section the concept of closest common derivator of queries.

DEFINITION 3.1. Let Q_1 and Q_2 be two queries and $\mathbf{R}_1$, $\mathbf{R}_2$ be two sets of relation occurrences from Q_1 and Q_2 respectively that have the same number of relation occurrences of each relation. A *common derivator* (CD) of Q_1 and Q_2 over the respective sets $\mathbf{R}_1$ and $\mathbf{R}_2$ is a view V such that there is a minimal rewriting of Q_1 (resp. Q_2) using V that involves V and only those relation occurrences of Q_1 (resp. Q_2) that do not appear in $\mathbf{R}_1$ (resp. $\mathbf{R}_2$.) □

Notice that queries Q_1 and Q_2 need not be defined over the same set of base relations, and their rewriting using V need not be complete. Clearly, V has as many relation occurrences of each base relation as they appear in $\mathbf{R}_1$ and $\mathbf{R}_2$.

EXAMPLE 3.1. Consider the relation schemas $U(F, A)$, $R(A, B, C)$, $S(C, D)$, and $T(D, E)$, and the queries

$Q_1 = \Pi_{R_1.B,\, R_2.A,\, R_3.C}(U \bowtie_{U.A \leq R_1.A} R_1[R] \bowtie_{R_1.B \leq R_2.A} \sigma_{B<3}(R_2[R]) \bowtie_{R_2.C \leq R_3.B} \sigma_{A \geq 4 \wedge A \leq 7}(R_3[R]) \bowtie_{R_3.C=S_1.C} \sigma_{D>3}(S_1[S]))$, and

$Q_2 = \Pi_{R_4.C,\, R_5.A,\, S_3.C}(S_2[S] \bowtie_{S_2.C \leq R_4.C} \sigma_{B=3}(R_4[R]) \bowtie_{R_4.C=R_5.B} \sigma_{A \geq 5 \wedge A \leq 9}(R_5[R]) \bowtie_{R_5.C \leq S_3.C} \sigma_{D \geq 3}(S_3[S]) \bowtie_{S_3.D.C=T.D} T)$,

Query Q_1 has three occurences of relation R and one occurrence of relation S, while query Q_2 has two occurrences of relation R and one occurrence of relation S. We use these schemas and queries as a running example in this section.

The view $V' = R_6[R] \bowtie_{R_6.C \leq R_7.B} R_7[R]$ is a CD of Q_1 and Q_2 over $\{R_2, R_3\}$ and $\{R_4, R_5\}$.

The views

$V'' = R_6[R] \bowtie_{R_6.C \leq R_7.B} \sigma_{A \geq 3 \wedge A \leq 9}(R_7[R]) \bowtie_{R_7.C \leq S_4.C} S_4[S]$,

$V''' = \sigma_{B \leq 3}(R_6[R]) \bowtie_{R_6.C \leq R_7.B} \sigma_{A \geq 3 \wedge A \leq 9}(R_7[R]) \bowtie_{R_7.C \leq S_4.C} \sigma_{D \geq 3}(S_4[S])$

$V = \Pi_{R_6.A,\, R_6.B,\, R_6.C,\, R_7.A,\, R_7.B,\, R_7.C,\, S_4.D}(\sigma_{B \leq 3}(R_6[R]) \bowtie_{R_6.C \leq R_7.B} \sigma_{A \geq 3 \wedge A \leq 9}(R_7[R]) \bowtie_{R_7.C \leq S_4.C} \sigma_{D \geq 3}(S_4[S]))$

are CDs of Q_1 and Q_2 over $\{R_2, R_3, S_1\}$ and $\{R_4, R_5, S_3\}$.

Both queries have minimal rewritings using each of the views V', V'', V''' and V. For instance, Q_1 and Q_2 can be minimally rewritten using V as follows:

$Q_1' = \Pi_{R_1.B,\, R_6.A,\, R_7.C}(U \bowtie R_1[R] \bowtie_{R_1.B \leq V_4.R_6.A} \sigma_{R_6.B<3 \wedge R_7.C=S_4.C \wedge R_7.A \geq 4 \wedge R_7.A \leq 7 \wedge S_4.D>3}(V))$, and

$Q_2' = \Pi_{R_4.C,\, R_7.A,\, S_4.C}(S_2[S] \bowtie_{S_2.C \leq V_4.R_6.C} \sigma_{R_6.C=R_7.B \wedge R_6.B=3 \wedge R_6.C=R_7.B \wedge R_7.A \geq 5 \wedge R_7.A \leq 9}(V) \bowtie T)$.

View V has two occurrences of R and one occurrence of S. Rewriting Q_1' has one occurrence of R, while rewriting Q_2' has one occurrence of S. □

A CD of two queries can be "closer" to the queries than some other CD of the same queries. We explain below this concept of closeness with an example before providing a formal definition. We use it then to define a closest common derivator of two queries. Roughly speaking, a closest common derivator of two queries is a CD that has as many relation occurrences as possible, and as few attributes and as few tuples in its answer as possible.

EXAMPLE 3.2. Consider the queries and the CDs of example 3.1.

- The CD $V'' = \sigma_{R_6.C \leq R_7.B \wedge R_7.A \geq 3 \wedge R_7.A \leq 9 \wedge R_7.C \leq S_4.C}(R_6[R] \times R_7[R] \times S_4[S])$ is closer to Q_1 and Q_2 than the CD $V' = \sigma_{R_6.C \leq R_7.B}(R_6[R] \times R_7[R])$ because:
  (a) V' is defined over the relation occurrence sets $\mathbf{R'}_1 = \{R_2, R_3\}$ of Q_1 and $\mathbf{R'}_2 = \{R_4, R_5\}$ of Q_2 while V'' is defined over the relation occurrence sets $\mathbf{R''}_1 = \{R_2, R_3, S_1\}$ of Q_1 and $\mathbf{R''}_2 = \{R_4, R_5, S_3\}$ of Q_2 and clearly $\mathbf{R'}_1 \subset \mathbf{R''}_1$ and $\mathbf{R'}_2 \subset \mathbf{R''}_2$.
  (b) V' is not more restrictive than V'' on their common relation occurrences (in fact, V' is less restrictive than V''.)

  Notice that the following conditions are satisfied:
  (a) $V'' \vdash V'$.
  (b) $\sigma_{R_6.C \leq R_7.B \wedge R_7.C \leq S_4.C}(R_6[R] \times R_7[R] \times S_4[S]) \not\Vdash \sigma_{R_6.C \leq R_7.B \wedge R_7.A \geq 3 \wedge R_7.A \leq 9 \wedge R_7.C \leq S_4.C}(R_6[R] \times R_7[R] \times S_4[S])$.

- The CD $V''' = \sigma_{R_6.B\leq 3 \wedge R_6.C\leq R_7.B \wedge R_7.A\geq 3 \wedge R_7.A\leq 9 \wedge R_7.C\leq S_4.C \wedge S_4.D\geq 3}(R_6[R] \times R_7[R] \times S_4[S])$ is closer to Q_1 and Q_2 than the CD V'' because:
  (a) V'' and V''' are defined over the same relation occurrence sets of Q_1 and Q_2.
  (b) V''' is more restrictive than V'' on their common relation occurrences.

  In this case the following conditions are satisfied:
  (a) $V''' \vdash V''$.
  (b) $\sigma_{R_6.B\leq 3 \wedge R_6.C\leq R_7.B \wedge R_7.A\geq 3 \wedge R_7.A\leq 9 \wedge R_7.C\leq S_4.C \wedge S_4.D\geq 3}(R_6[R] \times R_7[R] \times S_4[S]) \Vdash \sigma_{R_6.C\leq R_7.B \wedge R_7.A\geq 3 \wedge R_7.A\leq 9 \wedge R_7.C\leq S_4.C}(R_6[R] \times R_7[R] \times S_4[S])$.
  (c) $V'' \not\Vdash V'''$.
- The CD $V = \Pi_{R_6.A,\, R_6.B,\, R_6.C,\, R_7.A,\, R_7.B,\, R_7.C,\, S_4.D}(\sigma_{R_6.B\leq 3 \wedge R_6.C\leq R_7.B \wedge R_7.A\geq 3 \wedge R_7.A\leq 9 \wedge R_7.C\leq S_4.C \wedge S_4.D\geq 3}(R_6[R] \times R_7[R] \times S_4[S]))$ is closer to Q_1 and Q_2 than the CD V''' because:
  (a) V''' and V are defined over the same relation occurrence sets of Q_1 and Q_2.
  (b) V and V''' are equally restrictive on their common relation occurrences.
  (c) The set of projected attributes of V is a proper subset of that of V'''.

  The following conditions are satisfied:
  (a) $V \vdash V'''$.
  (b) $\sigma_{R_6.B\leq 3 \wedge R_6.C\leq R_7.B \wedge R_7.A\geq 3 \wedge R_7.A\leq 9 \wedge R_7.C\leq S_4.C \wedge S_4.D\geq 3}(R_6[R] \times R_7[R] \times S_4[S]) \Vdash \sigma_{R_6.B\leq 3 \wedge R_6.C\leq R_7.B \wedge R_7.A\geq 3 \wedge R_7.A\leq 9 \wedge R_7.C\leq S_4.C \wedge S_4.D\geq 3}(R_6[R] \times R_7[R] \times S_4[S])$.
  (c) $V''' \not\Vdash V$. □

In general, the names of relation occurrences in two CDs that are related with a closeness relationship may be distinct even if these CDs are defined over the same relation occurrence sets.

Definition 3.2. Let Q_1 and Q_2 be two queries, $V = \Pi_X(\sigma_C(\mathbf{R}))$ be a CD of Q_1 and Q_2 over $\mathbf{R}_1$ and $\mathbf{R}_2$, $V' = \Pi_{X'}(\sigma_{C'}(\mathbf{R}'))$ be a CD of Q_1 and Q_2 over $\mathbf{R}'_1$ and $\mathbf{R}'_2$, and let $\mathbf{R}_1 \subseteq \mathbf{R}'_1$ and $\mathbf{R}_2 \subseteq \mathbf{R}'_2$. The CD V' is *closer* to Q_1 and Q_2 than the CD V (denoted $V' \prec_{Q_1,Q_2} V$) if the following conditions are satisfied:
(a) $V' \vdash V$.
(b) If $\sigma_{C'}(\mathbf{R}') \Vdash \sigma_C(\mathbf{R})$ then $V \not\Vdash V'$. □

Clearly, two CDs of the same queries may not be related with a closeness relationship, even if they are defined over the same relation occurrence sets. We now formally define a closest common derivator of two queries.

Definition 3.3. Let Q_1 and Q_2 be two queries. A *closest common derivator* (CCD) of Q_1 and Q_2 over $\mathbf{R}_1$ and $\mathbf{R}_2$ is a CD V of Q_1 and Q_2 over $\mathbf{R}_1$ and $\mathbf{R}_2$ such that there exists no CD of Q_1 and Q_2 that is closer to Q_1 and Q_2 than V. □

Often, when referring to CDs and CCDs, we omit mentioning the relation occurrence sets over which they are defined, if this is not necessary for accuracy.

Example 3.3. Consider the queries Q_1 and Q_2 and the CDs V', V'', V''' and V of Example 3.1. The CDs V', V'' and V''' are not CCDs of Q_1 and Q_2 since, as shown in Example 3.2, for each of them there is a CD of Q_1 and Q_2 which is closer to Q_1 and Q_2. However, it can be shown that V is a CCD of Q_1 and Q_2. □

For queries related through a minimal rewriting the following proposition holds.

Proposition 3.1. Let Q_1 and Q_2 be two queries. If $Q_1 \vdash_m Q_2$ then Q_2 is a CCD of Q_1 and Q_2. □

Proposition 3.1 does not hold for a simple rewriting, even if this is a complete rewriting. As we saw in Example 2.4, it is possible that $Q_1 \not\vdash_m Q_2$ (in which case Q_2 cannot be a CCD of Q_1 and Q_2) even if $Q_1 \Vdash Q_2$.

Two queries may have several CCDs. The relation occurrence sets of these CCDs for each query may be disjoint, overlapping or contained.

Example 3.4. Consider the queries Q_1 and Q_2 of Example 3.1. In Example 3.3 we showed a CDD V of Q_1 and Q_2 over $\{R_2, R_3, S_1\}$ and $\{R_4, R_5, S_3\}$. It can be shown that view $U = \Pi_{S_5.C,\, S_5.D,\, R_8.A,\, R_8.B,\, R_8.C}(S_5[S] \bowtie_{S_5.C\leq R_8.C} R_8[R])$ is a CCD of Q_1 and Q_2 over $\{S_1, R_3\}$ and $\{S_2, R_4\}$ respectively. Set $\{S_1, R_3\}$ is contained into $\{R_2, R_3, S_1\}$, while set $\{S_2, R_4\}$ overlaps with $\{R_4, R_5, S_3\}$. □

In the next sections, we provide a method for computing all the CCDs of two queries along with minimal rewritings of these queries using each of the CCDs.

4. CONDITION MERGING

We show in this section how conditions can be merged. Merged conditions are used later to construct CCDs of queries.

A condition C is *consistent* if there is an assignment of values to its attributes that makes C true. A condition that is not consistent is called *inconsistent*. A condition C is *valid* if every assignment of values to its attributes makes C true. For instance, $A = A$ is a valid condition. Clearly, a condition is valid if it is a conjunction of valid atomic conditions. A condition C_1 *implies* a condition C_2 (notation $C_1 \models C_2$) if every assignment of values to the attributes of C_1 that makes C_1 true, also makes C_2 true. Two conditions C_1 and C_2 are *equivalent* (notation $C_1 \equiv C_2$) if $C_1 \models C_2$ and $C_2 \models C_1$. The complexity of the consistency and implication problems have been studied previously [20, 16, 27, 9, 10] for attributes ranging over dense totally ordered domains and for attributes ranging over the integers. Both problems have been shown to be polynomial for the class of conditions considered here.

A selection operation with an inconsistent condition returns an empty relation, while a selection operation with a valid condition returns all the tuples of the input relation. In the following, we assume that query conditions are consistent and not valid. Further, we assume that a query condition C does not include valid atomic conditions (otherwise, we can consider the condition resulting by removing from C all the valid atomic conditions which is a condition equivalent to C.)

Definition 4.1. Two conditions C_1 and C_2 are *mergeable* if there is a non-valid condition C such that $C_1 \models C$ and $C_2 \models C$ and there exists no condition C', $C' \not\equiv C$, such that $C_1 \models C'$, $C_2 \models C'$ and $C' \models C$. Condition C is called a *merge* of C_1 and C_2. □

Figure 1: (a) Conditions C_3 and C_4. (b) $merge(C_3, C_4)$

A merge of two conditions C_1 and C_2 is unique (otherwise, the conjunction of two non-equivalent merges C' and C'' of C_1 and C_2 is a merge of C_1 and C_2 that implies both C' and C''.) We denote the merge of C_1 and C_2 as $merge(C_1, C_2)$.

EXAMPLE 4.1. Let C_1 be the condition $B < D + 1.2 \wedge B \geq D$ and C_2 be the condition $B \leq D \wedge E < H$. The condition $C_1 \wedge C_2$ is consistent. It can be shown that C_1 and C_2 are mergeable and $merge(C_1, C_2)$ is $B < D + 1.2$.

Let C_3 be the condition $A > -5.1 \wedge A \leq 6$ and C_4 be the condition $A \geq 7 \wedge A < 9$. The condition $C_3 \wedge C_4$ is inconsistent. It can be shown that C_3 and C_4 are mergeable and $merge(C_3, C_4)$ is $A > -5.1 \wedge A < 9$. Conditions C_3 and C_4 and their merge are depicted on Figures 1(a) and (b).

Let C_5 be the condition $A > 5$ and C_6 be the condition $D = E + 3 \wedge B < 3$. The condition $C_5 \wedge C_6$ is consistent. It can be shown that no non-valid constraint can be implied by both C_5 and C_6, and therefore, C_5 and C_6 are not mergeable.

Let C_7 be the condition $A < 5$ and C_8 be the condition $A > 7$. The condition $C_7 \wedge C_8$ is inconsistent. It can be shown that C_7 and C_8 are not mergeable. □

The previous example shows that the mergeability of two conditions and the consistency of their conjunction are orthogonal properties. In the rest of this section we show how the merge of two conditions can be computed, starting with atomic conditions.

EXAMPLE 4.2. Let the atomic conditions A_1 and A_2 be $D > E$ and $D \geq E + 3$ respectively. Conditions A_1 and A_2 involve the same attributes D and E. They are depicted on Figure 2(a). Clearly, $A_2 \models A_1$. Conditions A_1 and A_2 are mergeable and $merge(A_1, A_2) = A_1$.

Let the atomic conditions A_3 and A_4 be $D > E$ and $E \leq 3.5$ respectively. Conditions A_3 and A_4 do not involve the same attributes: condition A_3 involves attributes D and E while condition A_4 involves only attribute E. Clearly, $A_3 \not\models A_4$ and $A_4 \not\models A_3$. One can see that A_3 and A_4 are not mergeable. □

More generally, we can show the proposition below. For this proposition we assume that atomic conditions do not involve equality (=).

PROPOSITION 4.1. Let A_1 and A_2 be two atomic conditions that do not involve equality. Conditions A_1 and A_2 are mergeable if and only if $A_1 \models A_2$ or $A_2 \models A_1$. If $A_1 \models A_2$, $merge(A_1, A_2) = A_2$. If $A_2 \models A_1$, $merge(A_1, A_2) = A_1$. □

As we show in Example 4.2 an atomic condition can imply another atomic condition that involves the same attributes. This is not the case for atomic conditions that do not involve the same attributes:

PROPOSITION 4.2. Let A_1 and A_2 be two atomic conditions that do not involve the same attributes. Then, $A_1 \not\models A_2$ and $A_2 \not\models A_1$. □

Thus, two atomic conditions can be mergeable only if they involve the same attributes.

Let us now consider general conditions. An atomic condition that involves equality can be equivalently replaced in a condition by the conjunction of two atomic conditions that involve $\leq$ and $\geq$. For instance, $D = E + 2$ can be replaced by $D \leq E + 2 \wedge D \geq E + 2$. An atomic condition A_1 in a condition C is *strongly redundant* in C, if there is another atomic condition A_2 in C such that $A_2 \models A_1$. For instance, $D < E + 2$ is strongly redundant in a condition C if the atomic condition $D < E$ is also present in C. Removing from a condition C an atomic condition that is strongly redundant in C results in a condition equivalent to C. Thus, without loss of generality, we can consider that conditions do not involve equality and do not include strongly redundant atomic conditions.

In order to determine the merge of two conditions we need a form for conditions that directly represents the atomic formulas that can be implied by the conditions.

DEFINITION 4.2. A condition C is in *full* form if:

1. For every atomic condition A_i such that $C \models A_i$, there is an atomic condition A_j in C such that $A_j \models A_i$.
2. Condition C does not include strongly redundant atomic conditions.

A query $\Pi_X(\sigma_C(\mathbf{R}))$ is in *full* form if its condition C is in full from. □

Every condition can be equivalently put in full form in polynomial time by computing the closure of its atomic conditions [16, 27, 9] and removing strongly redundant atomic conditions. Note that a condition C in full form may include redundant atomic conditions, that is, atomic conditions that can be implied by the conjunction of the rest of the atomic conditions in C.

The following therorem determines whether two conditions are mergeable and shows how their merge can be computed. By convention, we assume that if two conditions C_1 and C_2 are not mergeable, $merge(C_1, C_2) = T$, where T denotes the truth value TRUE.

THEOREM 4.1. Let C_1 and C_2 be two conditions in full form that do not involve equality. Let

$$\mathcal{M} = \bigwedge_{A_i \text{ in } C_1,\ A_j \text{ in } C_2} merge(A_i, A_j).$$

where A_i and A_i denote atomic conditions. If $\mathcal{M} = T$, conditions C_1 and C_2 are not mergeable. Otherwise, C_1 and C_2 are mergeable and $\mathcal{M} = merge(C_1, C_2)$. □

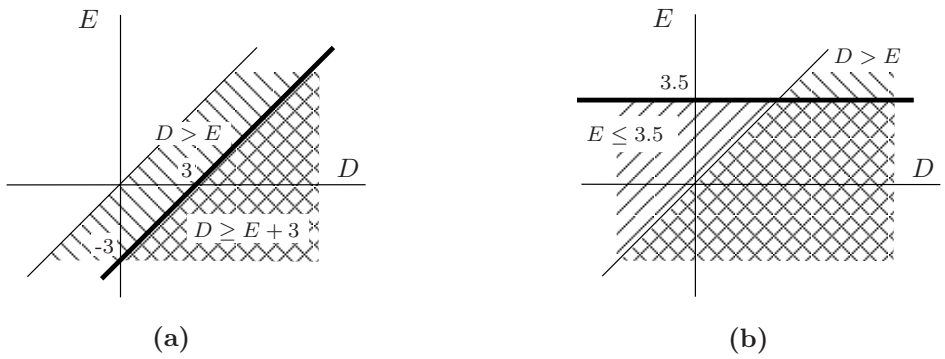

Figure 2: (a) Mergeable atomic conditions. (b) Non-mergeable atomic conditions.

EXAMPLE 4.3. Applying Proposition 4.1 to the conditions of Example 4.1 validates the claims made there. □

The merge of two conditions can be computed in polynomial time. Usually the number of atomic conditions in a query is not large. Therefore, the merging of conditions can be performed efficiently even for a large number of queries.

5. COMPUTING CCDS USING QUERY GRAPHS

We show in this section how queries can be represented using graphs called query graphs. Then, we provide necessary and sufficient conditions for a view to be a CCD of two queries in terms of query graphs. These conditions are used to compute all the CCDs of two queries.

DEFINITION 5.1. Given a query $Q = \Pi_X(\sigma_C(\mathbf{R}))$, a *query graph* for Q is a node and edge labeled graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ where:

1. $\mathcal{N}$ is the set of nodes of $\mathcal{G}$. Set $\mathcal{N}$ comprises exactly one node for every relation occurrence in $\mathbf{R}$. Every node is labeled by:
   (a) the name R of the represented relation occurrence and the name of the corresponding relation if this last one is renamed, and
   (b) the set P_R of the projected attributes of the represented relation occurrence (those attributes of this relation occurrence that appear in X.)

   Nodes in $\mathcal{G}$ can be uniquely identified by the name of the represented relation occurrence. In the following we identify a node in $\mathcal{G}$ with the represented relation occurrence.
2. $\mathcal{E}$ is the set of edges of $\mathcal{G}$. For every node R in $\mathcal{N}$, if R has an attribute involved in a selection atomic condition in C, there is a loop edge $\langle R, R\rangle$ in $\mathcal{E}$ labeled by the conjunction C_R of all the selection atomic conditions in C involving attributes of R. For every two nodes R, S in $\mathcal{N}$ that have a join atomic condition in C involving an attribute of R and an attribute of S, there is an edge $\langle R, S\rangle$ in E labeled by the conjunction C_{RS} of all the join atomic conditions in C involving one attribute of R and one attribute of S. The conditions C_R and C_{RS} are called *edge conditions* of the edges $\langle R, R\rangle$ and $\langle R, S\rangle$ respectively. □

EXAMPLE 5.1. Figure 3 shows the query graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ for the queries Q_1 and Q_2 of Example 3.1. The names of the nodes are shown by the nodes. The projected attributes of a node follow the name of the node separated by colon. Edge conditions are shown by the corresponding edges. □

In the following, we identify a query with its query graph. We now define the concept of a candidate CCD of two query graphs. This definition uses node mapping functions between two query graphs. If C is a condition, and f is a node mapping function that maps all the relation occurrences (nodes) of C, we denote by $f(C)$ the condition obtained by renaming the relation occurrences in C according to the relation renaming induced by f. Further, if $R_i.A$ is an attribute of the relation occurrence R_i, and f is a node mapping function that maps R_i to a relation occurrence R_j, we denote by $f(R_i.A)$ the attribute $R_j.A$.

DEFINITION 5.2. Let $\mathcal{G}_1 = (\mathcal{N}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{N}_2, \mathcal{E}_2)$ be the query graphs of two queries in full form, and $\mathbf{R}_1$ and $\mathbf{R}_2$ be two sets of nodes in $\mathcal{G}_1$ and $\mathcal{G}_2$ respectively ($\mathbf{R}_1 \subseteq \mathcal{N}_1$ and $\mathbf{R}_2 \subseteq \mathcal{N}_2$). A query graph $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ is a *candidate CCD* of $\mathcal{G}_1$ and $\mathcal{G}_2$ over $\mathbf{R}_1$ and $\mathbf{R}_2$ if and only if there are two one-to-one onto functions f_1 and f_2 from $\mathcal{N}$ to $\mathbf{R}_1$ and from $\mathcal{N}$ to $\mathbf{R}_2$ respectively such that:

1. For every node $R \in \mathcal{N}$, the nodes R, $f_1(R)$ and $f_2(R)$ are relation occurrences of the same relation.
2. For every $\langle R, S\rangle \in \mathcal{E}$, $\langle f_i(R), f_i(S)\rangle \in \mathcal{E}_i$, $i = 1, 2$. That is, edges in $\mathcal{G}$ are mapped through f_i, $i = 1, 2$, to edges in $\mathcal{G}_i$.
3. For every condition C of an edge $\langle R, S\rangle \in \mathcal{E}$, the condition $f_1^{-1}(C_1)$, where C_1 is the condition of the edge $\langle f_1(R), f_1(S)\rangle$, and the condition $f_2^{-1}(C_2)$, where C_2 is the condition of the edge $\langle f_2(R), f_2(S)\rangle$, are mergeable and $C = merge(f_1^{-1}(C_1), f_2^{-1}(C_2))$. That is, each condition of an edge in $\mathcal{G}$ is the merge of the conditions of the images of this edge in $\mathcal{G}_1$ and $\mathcal{G}_2$ under the relation occurrence renamings induced by the inverse of the node mapping functions.
4. For every node $R \in \mathcal{N}$, its set of projected attributes is $\bigcup_{i=1,2}(Y_i \cup Z_i)$, where:
   (a) Y_i is the set of attributes $R.A$ such that $f_i(R.A)$ is a projected attribute of $f_i(R)$ in $\mathcal{G}_i$, and
   (b) Z_i is the set of attributes $R.A$ such that attribute $f_i(R.A)$ is involved in an atomic condition A_i of the condition of an edge $\langle f_i(R), f_i(S)\rangle$ of $\mathcal{G}_i$ and $C \not\models f_i^{-1}(A_i)$, where C is the condition of the edge $\langle R, S\rangle$ of $\mathcal{G}$. Edge $\langle f_i(R), f_i(S)\rangle$ can be a loop edge, i.e. $f_i^{-1}(S)$ can be identical to $f_i^{-1}(R)$.

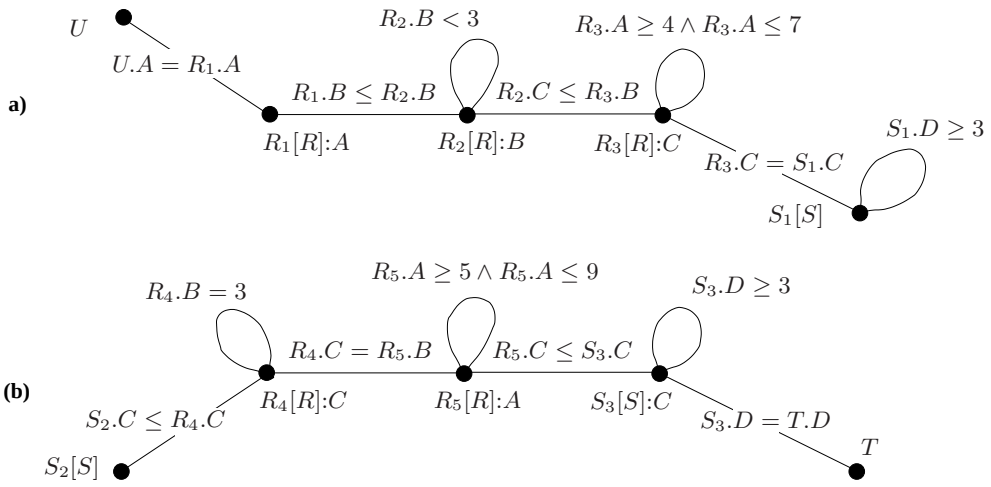

Figure 3: (a) Query graph $\mathcal{G}_1$. (b) Query graph $\mathcal{G}_2$

5. There is no query graph $\mathcal{G}' = (\mathcal{N}', \mathcal{E}')$, and extensions $f'_i,\ i = 1, 2$, of f_i from $\mathcal{N}'$ to $\mathbf{R}'_i$, where $\mathbf{R}_i \subset \mathbf{R}'_i \subseteq \mathcal{N}_i$, that satisfy the properties 1 - 4 above. □

EXAMPLE 5.2. Figure 4 shows two candidate CCDs $\mathcal{G}$ and $\mathcal{G}'$ of the query graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ of Example 5.1. Query graph $\mathcal{G}$ on Figure 4(a) is a CCD of $\mathcal{G}_1$ and $\mathcal{G}_2$ over the set of nodes $\{R_2, R_3, S1\}$ and $\{R_4, R_5, S_1\}$. Query graph $\mathcal{G}'$ on Figure 4(b) is a CCD of $\mathcal{G}_1$ and $\mathcal{G}_2$ over the set of nodes $\{S_1, R_3\}$ and $\{S_2, R_4\}$. □

Clearly, a candidate CCD of two query graphs is a CD of these query graphs. Next we show how a closeness relationship between two candidate CCDs can be expressed in terms of query graphs.

PROPOSITION 5.1. Let $\mathcal{G}_1 = (\mathcal{N}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{N}_2, \mathcal{E}_2)$ be the query graphs of two queries in full form, and $\mathbf{R}_1$, $\mathbf{R}_2$, $\mathbf{R}'_1$ and $\mathbf{R}'_2$ be sets of nodes such that $\mathbf{R}_1 \subseteq \mathbf{R}'_1 \subseteq \mathcal{N}_1$ and $\mathbf{R}_2 \subseteq \mathbf{R}'_2 \subseteq \mathcal{N}_2$. Let also $\mathcal{G} = (\mathcal{N}, \mathcal{E})$ be a candidate CCD of $\mathcal{G}_1$ and $\mathcal{G}_2$ over $\mathbf{R}_1$, and $\mathbf{R}_2$ and $\mathcal{G}' = (\mathcal{N}', \mathcal{E}')$ be a candidate CCD of $\mathcal{G}_1$ and $\mathcal{G}_2$ over $\mathbf{R}'_1$ and $\mathbf{R}'_2$. $\mathcal{G}' \prec_{\mathcal{G}_1,\mathcal{G}_2} \mathcal{G}$ if and only if $\mathcal{G}$ and $\mathcal{G}'$ are not equivalent and there is a one-to-one node mapping function f from $\mathcal{G}$ to $\mathcal{G}'$ such that:

1. For every node $R \in \mathcal{N}$, the nodes R and $f(R)$ are relation occurrences of the same relation.
2. For every $\langle R, S\rangle \in \mathcal{E}$, $\langle f(R), f(S)\rangle \in \mathcal{E}'$.
3. For every condition C of an edge $\langle R, S\rangle \in \mathcal{E}$, $f(C) \models C$, where $f(C)$ is the condition of the edge $\langle f(R), f(S)\rangle \in \mathcal{E}'$.
4. If f is an onto function (that is, if $\mathcal{G}$ and $\mathcal{G}'$ have the same number of nodes), for every node $R \in \mathcal{N}$, $P_R \supseteq P_{R'}$, where P_R is the set of projected attributes of node R and $P_{R'}$ is the set of projected attributes of node $R' = f(R)$. □

The following theorem provides necessary and sufficient conditions for a query graph to be a CCD of two query graphs.

THEOREM 5.1. A query graph $\mathcal{G}$ is a CCD of two query graphs $\mathcal{G}_1$ and $\mathcal{G}_2$ over $\mathbf{R}_1$ and $\mathbf{R}_2$ if and only if $\mathcal{G}$ is a candidate CCD of $\mathcal{G}_1$ and $\mathcal{G}_2$ over $\mathbf{R}_1$ and $\mathbf{R}_2$, and there exists no candidate CCD $\mathcal{G}'$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ over $\mathbf{R}'_1$ and $\mathbf{R}'_2$, where $\mathbf{R}_1 \subseteq \mathbf{R}'_1$ and $\mathbf{R}_2 \subseteq \mathbf{R}'_2$, such that $\mathcal{G}' \prec_{\mathcal{G}_1,\mathcal{G}_2} \mathcal{G}$. □

The previous results suggest a process for computing all the CCDs of two queries which is outlined below.

Input: two query graphs $\mathcal{G}_1 = (\mathcal{N}_1, \mathcal{E}_1)$ and $\mathcal{G}_2 = (\mathcal{N}_2, \mathcal{E}_2)$.
Output: the set C of all the CCDs of $\mathcal{G}_1$ and $\mathcal{G}_2$

1. Compute the set C of all the candidate CCDs of $\mathcal{G}_1$ and $\mathcal{G}_2$ by identifying node mapping functions f from $\mathcal{N}_1$ to $\mathcal{N}_2$ that map an edge $\langle R_1, S_1\rangle \in \mathcal{E}_1$ to an edge $\langle R_2, S_2\rangle \in \mathcal{E}_2$ such that the condition $f(C_1)$, where C_1 is the condition of $\langle R_1, S_1\rangle$, and the condition C_2 of $\langle R_2, S_2\rangle$ are mergeable.
2. Remove from $\mathcal{C}$ a candidate CCD $\mathcal{G}$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ if there is a candidate CCD $\mathcal{G}'$ of $\mathcal{G}_1$ and $\mathcal{G}_2$ such that $\mathcal{G}' \prec_{\mathcal{G}_1,\mathcal{G}_2} \mathcal{G}$.□

The previous process can be extended in a straightforward way to generate also minimal rewritings of the queries using the computed CCDs based on the identified node mapping functions.

6. CONCLUSION

We have addressed the problem of constructing a search space for materialized view selection. This is an intricate problem since it requires the detection and exploitation of common subexpressions between queries and views. We have suggested a novel approach which is based on adding to the alternative plans of the queries CCDs, and on rewriting the queries using these CCDs. CCDs are defined using the query definitions and do not depend on a specific query evaluation plan. CCDs generalize previous definitions of common subexpressions since two queries can be partially rewritten using their CCD. Adding CCDs to the alternative evaluation plans of the queries generates a search space which is expected to comprise all the interesting views for materialization since the CCD of two queries is as close to these queries as possible. A traditional query optimizer can be used to generate alternative evaluation plans for a CCD. The existence of a CCD in the search space does not force its materialization by the view selection algorithms, nor does it

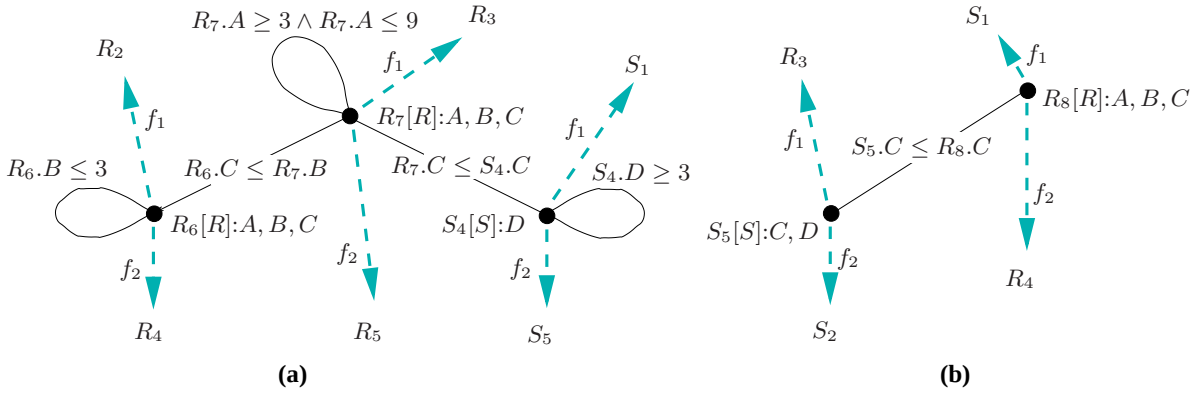

Figure 4: Two CCDs of query graphs $\mathcal{G}_1$ and $\mathcal{G}_2$.

exclude the materialization of other nodes (views) in any of its alternative evaluation plans.

We have formally defined CCDs based on a closeness relationship on CDs. Using a declarative query graph representation for queries we provided necessary and sufficient conditions for a view to be a CCD of two queries. Based on these results we have outlined a process for computing all the CCDs of two queries and for generating minimal rewritings of the queries using their CCDs.

We are currently working towards two directions. The first one involves extending our results to a more general class of queries and views. In particular, we are examining how the concepts presented here can be extended to queries involving grouping and aggregation operations which are common in data warehousing applications. The second one concerns the design of view selection algorithms which can heuristically decide which CCDs need to be generated based on a cost model and on the requirements the Data Warehouse needs to satisfy. This will avoid the construction of the complete search space during the view selection process which can be unfeasible for a large number of queries that share common subexpresions.

7. REFERENCES

[1] M. O. Akinde and M. H. Böhlen. Constructing GPSJ View Graphs. In *Proc. of the Intl. Workshop on Design and Management of Data Warehouses, Germany*, pages 8/1–12, 1999.

[2] E. Baralis, S. Paraboshi, and E. Teniente. Materialized view selection in a multidimensional database. In *Proc. of the 23rd Intl. Conf. on Very Large Data Bases*, pages 156–165, 1997.

[3] U. S. Chakravarthy and J. Minker. Multiple Query Processing in Deductive Databases Using Query Graphs. In *Proc. of the 12th Intl. Conf. on Very Large Data Bases*, pages 384–391, 1986.

[4] F.-C. F. Chen and M. H. Dunham. Common subexpression processing in multiple-query processing. *IEEE Transactions on Knowledge and Data Engineering*, 10(3):493–499, 1998.

[5] S. Cohen, W. Nutt, and A. Serebrenik. Rewriting Aggregate Queries Using Views. In *Proc. of the 18th ACM Symp. on Principles of Database Systems*, pages 155–166, 1999.

[6] S. Finkelstein. Common Expression Analysis in Database Applications. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 235–245, 1982.

[7] M. Golfarelli and S. Rizzi. Comparing Nested GPSJ Queries in Multidimensional Databases. In *Proc. of the 3rd Intl. Workshop on Data Warehousing and OLAP*, pages 65–71, 2000.

[8] M. Golfarelli and S. Rizzi. View Materialization for Nested GPSJ Queries. In *Proc. of the Intl. Workshop on Design and Management of Data Warehouses*, pages 10/1–9, 2000.

[9] S. Guo, W. Sun, and M. A. Weiss. On Satisfiability, Equivalence, and Implication Problems Involving Conjunctive Queries in Database Systems. *IEEE Transactions on Knowledge and Data Engineering*, 8(4):604–616, 1996.

[10] S. Guo, W. Sun, and M. A. Weiss. Solving Satisfiability Problems in Database Systems. *ACM Transactions on Database Systems*, 21(2):270–293, 1996.

[11] H. Gupta, V. Harinarayan, A. Rajaraman, and J. D. Ullman. Index Selection for OLAP. In *Proc. of the 13th Intl. Conf. on Data Engineering*, pages 208–219, 1997.

[12] H. Gupta and I. S. Mumick. Selection of Views to Materialize Under a Maintenance Cost Constraint. In *Proc. of the 7th Intl. Conf. on Database Theory*, pages 453–470, 1999.

[13] V. Harinarayan, A. Rajaraman, and J. D. Ullman. Implementing Data Cubes Efficiently. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, 1996.

[14] M. Jarke. Common Subexpression Isolation in Multiple Query Optimization. In W. Kim, D. S. Reiner, and D. S. Batory, editors, *Query Processing in DB Systems*, pages 191–205. Springer, 1985.

[15] P. Kalnis, N. Mamoulis, and D. Papadias. View Selection Using Randomized Search. *Data and Knowledge Engineering*, 42(1):89–111, 2002.

[16] A. Klug. On Conjunctive Queries Containing Inequalities. *Journal of the ACM*, 35(1):146–160, 1988.

[17] Y. Kotidis and N. Roussopoulos. DynaMat: A Dynamic View Management System for Data Warehouses. In *Proc. of the ACM SIGMOD Intl. Conf. on Management of Data*, pages 371–382, 1999.

[18] M. Lee and J. Hammer. Speeding Up Materialized View Selection in Data Warehouses Using a Randomized algorithm. *International Journal of Cooperative Information Systems*, 10(3):327–253, 2001.

[19] A. Levy, A. O. Mendelson, Y. Sagiv, and D. Srivastava. Answering Queries using Views. In *Proc. of the ACM Symp. on Principles of Database Systems*, pages 95–104, 1995.

[20] D. J. Rosenkrantz and H. B. Hunt. Processing Conjunctive Predicates and Queries. In *Proc. of the Intl. Conf. on Very Large Data Bases*, pages 64–72, 1980.

[21] T. K. Sellis. Multiple Query Optimization. *ACM Transactions on Database Systems*, 13(1):23–52, 1988.

[22] A. Shukla, P. M. Deshpande, and J. F. Naughton. Materialized View Selection for Multidimensional Datasets. In *Proc. of the 24d Intl. Conf. on Very Large Data Bases*, pages 488–499, 1998.

[23] D. Theodoratos and M. Bouzeghoub. A General Framework for the View Selection Problem for Data Warehouse Design and Evolution. In *Proc. of the 3rd Intl. Workshop on Data Warehousing and OLAP*, pages 1–9, 2000.

[24] D. Theodoratos, S. Ligoudistianos, and T. Sellis. View Selection for Designing the Global Data Warehouse. *Data and Knowledge Engineering, Elsevier Science*, 39(3):219–240, 2001.

[25] D. Theodoratos and T. Sellis. Data Warehouse Configuration. In *Proc. of the 23rd Intl. Conf. on Very Large Data Bases*, pages 126–135, 1997.

[26] D. Theodoratos and T. Sellis. Incremental Design of a Data Warehouse. *Journal of Intelligent Information Systems, Kluwer Academic Publishers*, 15(1):7–27, 2000.

[27] J. D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume 2. Computer Science Press, 1989.

[28] J. Yang, K. Karlapalem, and Q. Li. Algorithms for Materialized View Design in Data Warehousing Environment. In *Proc. of the 23rd Intl. Conf. on Very Large Data Bases*, pages 136–145, 1997.