Conjunctive Query Rewriting in the Context of Data Integration

Kang-Sik Moon and Jeon-Young Lee

IIS Lab., CSE Dept., POSTECH
San31, Hyoja-dong, Nam-gu, Pohang, Kyungbuk, South Korea
Tel: +82-54-279-5662, Fax: +82-54-279-8498, E-mail: {ksmoon, jeon}@postech.ac.kr

Abstract

The problem of query rewriting using views has interested in the context of data integration where source data is described by the views on global relations. When the query and views are of the form of conjunctive queries, the rewriting is a union of conjunctive queries each of which is contained in the original query and consists of only views. Most previous methods for query rewriting using views are 2-step algorithms. In the first step, they identify the views that are useful in rewriting and in the second step they construct all correct rewritings by combining the views that gained in the first step. The larger the number of selected views in the first step, the larger the number of candidate rewritings in the second step. We want to minimize the number of selected views in the first step by defining stringent conditions for a view to be participated in rewritings. In this paper, first we offer a necessary condition for the existence of a rewriting that includes a view. For the common case that predicate repetitions are not allowed in the bodies of views, we show that our algorithm for testing the condition is done in a polynomial-time. Second, we offer an algorithm to construct contained rewritings using the view instances that are computed in the first step. The exponential containment-mapping test in the second step is not needed in our algorithm.

Keywords:

Query containment; Query rewriting; Contained rewriting; Data integration

1. Introduction

Recently, the problem of query rewriting using views has received a lot of interest in the fields of AI and Database systems; we refer to the survey by Levy for background [5]. In the context of conventional query optimization, the equivalent rewriting is considered; that is, the solution of the problem is to find equivalent rewritings of the user query. In contrast to the case of query optimization, in the context of data integration where data sources are considered to store materialized views over a global database schema [13], we usually cannot find a rewriting that is equivalent to the user query because of the data source's limited coverage. Instead, we search for all *contained rewritings*, each of which is contained in the user query, given the available sources.

Levy [6] showed that all possible equivalent rewritings of a query can be obtained by considering containment mappings [2] from the bodies of the views to the body of the qeury. They also showed that the problem was NP-completeness for conjunctive queries with no built-in predicates. But their result is not appropriate in the contained rewriting problem. In the case of the contained rewriting, the views can have predicates that do not appear in the body of the query so that there are no containment mappings from the bodies of the views to the body of query, but a rewriting using the views can exist.

Most previous algorithms for the contained rewriting problem are 2-step algorithms. In the first step, they identify the views that are *useful* in rewriting. In the second step, they construct all sound rewritings by combining the views that are gained in the first step. The larger the number of selected views in the first step, the larger the number of candidate rewritings in the second step. We want to minimize the number of selected views in the first step by defining stringent condition for a view to participate in the rewritings.

In this paper, we show that the contained rewriting problem is closely related to some containment test, called *filter containment*, between a view and some partial queries of the user query. By testing the containment, the role of a view in some rewritings can be determined earlier, thereby eliminating useless views from the search space of the second step is possible. We show that for the common case when predicate repetitions are not allowed in the bodies of views, the containment test is done in polynomial-time. In chapter 3, we introduce the filter containment problem that is the theoretical basis of the contained rewriting problem. In chapter 4, we describe our query-rewriting algorithm using the filter containment.

2. Preliminaries

Views and user queries are represented using the logical rule notation described in [12]. A rule has the form

$q(\overline{X})$:- $r_1(\overline{X}_1), \ldots, r_k(\overline{X}_k)$

where q, and $r_1, \ldots, r_k$ are predicate names and $\overline{X}$, $\overline{X}_1$, ..., $\overline{X}_k$ are tuples of variables or constants. The head of the rule is $q(\overline{X})$. The body of the rule is the conjunction of subgoals $r_1(\overline{X}_1), \ldots, r_k(\overline{X}_k)$. A variable in $\overline{X}$ is called a *distinguished variable.* A variable that occurs in the body and is not a distinguished variable is called an *existential variable* (or non-distinguished variable). A *shared variable* is a variable that appears in more than one subgoal. A subgoal in the body of the rule represents a logical database relation. We use $\Lambda = V_1, .., V_n$ to denote views that are defined over the database relations. We assume that all variables in a query and views initially have distinct names, so that no two views (or query and view) share the same variable. We also assume that all rules are safe.

Let s and t represent two arbitrary predicate occurrences with the same name. We say that there is a *predicate mapping* $s \to t$ if there is a symbol mapping for the symbols in s under which s is made syntactically identical to t; such symbol mapping is termed the *symbol mapping induced by the predicate mapping.* In a predicate mapping $s \to t$, t is the destination of s under it. Two symbol mappings are said to be *consistent* if no variable is assigned a different value and constant is mapped to only the constant by the two symbol mappings, and a pair of predicate mappings is said to be *consistent* if the symbol mappings induced by the mappings are consistent.

Conventionally, a query C_2 is contained in C_1 if there is a containment mapping h: $C_1 \to C_2$ [2]. In this paper, we define containment using predicate mappings as follows:

Definition 2.1. (Query containment) Consider two queries.

C_1: $a(\overline{X})$:- $a_1(\overline{X}_1), \ldots, a_k(\overline{X}_k)$.

C_2: $b(\overline{Y})$:- $b_1(\overline{Y}_1), \ldots, b_l(\overline{Y}_l)$.

A query C_1 *contains* a query C_2 if and only if:

1. For all i and some j, there is a predicate mapping $a_i \to b_j$ and each pair of predicate mappings is consistent and,
2. All distinguished variable in C_1 maps to a distinguished variable in C_2 under the union of symbol mappings induced by the predicate mappings described in 1. □

Then the conventional containment mapping h: $C_1 \to C_2$ may be defined as the union of symbol mappings induced by the predicate mappings $a_i \to b_j$, for all i. (Note that the conventional containment mapping must map the head of C_1 to the head of C_2; however, in this paper we test just whether $h(\overline{X}) \subset \overline{Y}$).

Let R be a query over Λ, and consider a subgoal $v(\overline{X})$ of R, an instance of a view $V \in \Lambda$. By the *expansion* of this subgoal $Exp(v(\overline{X}))$, we mean the body of V, in which we rename the distinguished variables of V by the corresponding variables in $\overline{X}$, and retain the other variables. The *expansion* of R then is a query E over the database relations obtained as follows. The head of E equals that of R. The body of E is formed by including an expansion of each subgoal of R.

Definition 2.2. (Contained rewriting) Given a query Q and views Λ, a query R over Λ is a *contained rewriting* if the expansion of R is contained in Q. A rewriting R, with expansion E, is a *maximally contained rewriting* if (1) R is a contained rewriting of Q, and (2) for any other rewriting R' of Q, with expansion E', if E' is contained in E. □

In this paper, we discuss an algorithm to obtain a maximally contained rewriting of a query using views. In an actual, the maximally contained rewriting is a union of all possible contained rewritings. We say that a view v *covers* a query subgoal g in a contained rewriting if $g \to p$ appears in the set of predicate mappings that result in the containment mapping from the query to the expansion of the rewriting, where p is a subgoal of $Exp(v)$. In this paper, we consider only the contained rewritings whose views cover at least more than one query subgoal.

Example 2.1. Consider the following query Q and views.

Q : q(x,w) :- $p1$(x,y), $p2$(y,z), $p3$(z,'A',w).

V_1 : v_1(a1,c1,d1) :- $p1$(a1,b1), $p2$(b1,c1), $p3$(d1,e1,f1).

V_2 : v_2(a2,b2,c2) :- $p3$(a2,b2,c2), $p4$(c2,d2).

V_3 : v_3(b3,c3,d3,e3) :- $p2$(a3,b3), $p3$(c3,d3,e3), $p4$(a3,f3).

Then, there are two contained rewritings R_1 and R_2 like followings.

R_1 : q(x,w) :- v_1(x,z,d1), v_2(z,'A',w).

R_2 : q(x,w) :- v_1(x,z,d1), v_3(b3,z,'A',w).

We have the following expansions of the corresponding rewritings:

E_1 : q(x,w) :- $p1$(x,b1), $p2$(b1,z), $p3$(d1,e1,f1), $p3$(z,'A',w), $p4$(w,d2).

E_2 : q(x,w) :- $p1$(x,b1), $p2$(b1,z), $p3$(d1,e1,f1), $p2$(a3,b3), $p3$(z,'A',w), $p4$(a3,f3).

We know that the rewriting R_1 is sound by considering the containment mapping from Q to the expansion E_1 {x→x, y→b1, z→z, 'A'→'A', w→w}; it is obtained from the union of the symbol mappings induced by the forced predicated mappings $p1$(x,y)→$p1$(x,b1), $p2$(y,z)→$p2$(b1,z), and $p3$(z,'A',w)→$p3$(z,'A',w). In the case of R_2, the containment mapping is {x→x, y→b1, z→z, 'A'→'A', w→w}. Then, the maximally contained rewriting of Q is $R_1 \cup R_2$.□

In the above example, v_1(x,z,d1) covers the query subgoals $p1$(x,y) and $p2$(y,z) in the rewriting R_1, because the subgoals map to the view subgoals $p1$(x,b1) and $p2$(b1,z), respectively, which are in the expansion of v_1(x,z,d1). In the same way, v_2(z,'A',w) covers the query subgoal $p3$(z,'A',w) in the rewriting R_1, because the subgoal map to the subgoal $p3$(z,'A',w), which is in the expansion of v_2(z,'A',w).

Given a query Q and a subset of query subgoals G, the *partial* query of Q on G is a query obtained as follows. The body is a conjunction of subgoals in G. The head variables consist of only distinguished query variables in G and non-distinguished shared query variables that appears in G as well as in any other subgoals of Q. Semantically, the result of the partial query on G is a minimal set of tuples that is needed from the subgoals G to compute the user query.

The intuition behind our solution for the contained rewriting problem is the following. Suppose Q: $q()$:- $r_1()$, ..., $r_n()$ is the user query defined on database relations. Then there are many equivalent queries of the form Q': $q()$:- $p_1()$, ..., $p_m()$ where each $p_i()$ is the head of the partial query on G_i, $G_1 \cup .. \cup G_m$ = Subgoals(Q), and for all $i \neq j$, $G_i \cap G_j = \emptyset$. The contained rewriting corresponding to each Q' can be obtained by combining the views each of which replaces one of the partial queries so that all partial queries in Q' is replaced with the views. Intuitively, a partial query p_i can be replaced with a view if the view provides *necessary* tuples from the set G_i to compute the answer of Q'. In the case of equivalent rewriting, the necessary tuples are the same as the result of p_i while in the case of contained rewriting the necessary tuples are the subset of the result of p_i. Therefore, a view can replace a partial query in a rewriting if a new query can be constructed from the view such that the new query is contained in the partial query. Views that cannot replace any partial query are useless for rewriting. The example below shows the observation underlying our solution for the problem.

Example 2.2. Suppose that in addition to the query and the views of example 2.1 we also consider the following partial queries of Q:

P_{12} : q_{12}(x,z) :- $p1$(x,y), $p2$(y,z).

P_3 : q_3(z,'A',w) :- $p3$(z,'A',w).

Note that the partial queries P_{12} and P_3 can be easily obtained from the query Q by forcing the necessary projections and selections on the sets of subgoals {$p1$, $p2$} and {$p3$}, respectively, to compute the query. Then we know easily that the following query Q' is equivalent to the query Q:

Q' : q(x,w) :- q_{12}(x,z), q_3(z,'A',w).

Let's consider the relationship among the partial queries and the modified views $\sigma_\emptyset(V_1)$ and $\sigma_{(b='A')}(V_2)$. We know that $\sigma_\emptyset(V_1)$ is contained in P_{12} by considering the containment mapping h_{12}: $P_{12} \rightarrow \sigma_\emptyset(V_1)$ = $p1$(x,y)$\rightarrow p1$(a1,b1) $\cup$ $p2$(y,z)$\rightarrow p2$(b1,c1) = {x$\rightarrow$a1, y$\rightarrow$b1, z$\rightarrow$c1}. Similarly, $\sigma_{(b='A')}(V_2)$ is contained in P_3 and the containment mapping h_3: $P_3 \rightarrow \sigma_{(b='A')}(V_2)$ is $p3$(z,'A',w)$\rightarrow p3$(a3,'A',c3) = {z$\rightarrow$a3, 'A'$\rightarrow$'A', w$\rightarrow$c3}. On our intuition, because V_1 and V_2 provide the subset of the result of P_{12} and P_3 respectively, V_1 and V_2 can replace q_{12}(x,z) and q_3(z,'A',w) in Q' respectively. The view instance v_1(x,z,d1) in the rewriting R_1 can be obtained by applying the inverse of the containment mapping h_{12} to the head of $\sigma_\emptyset(V_1)$. Similarly, the view instance v_2(z,'A',w) in the rewriting by applying the inverse of the containment mapping h_3 to the head of $\sigma_{(b='A')}(V_2)$.□

3. Filter containment Problem

In this chapter, first, we define modified containment, termed *filter containment*, between a partial query of the user query and a view that will be used to determine whether a new query can be constructed from the view such that the new query is contained in the partial query. Next, we describe an algorithm for testing the filter containment. Finally, we show that if a view in a rewriting covers a subset of query subgoals, then the view is contained in the partial query over the subset of query subgoals.

Definition 3.1. (Filtered view) Given a view V, a filter σ on V is a set of equality predicates on the distinguished variables of V. Then the *filtered* view of V under σ is a query $\sigma(V)$ obtained by applying all predicates in σ to the head and body of V. □

Example 3.1. Consider a view V_4: v_4(a,b,c,d) :- $p2$(a,b), $p3$(c,d,e), $p4$(a,f) and a filter σ = {c=b, d='A'}. We know that σ is applicable to V_4 because all variables in σ are the distinguished variables of V_4. Then the following expression is the filtered view of V_4 under σ.

$\sigma(V_4)$: v_4(a,b,b,'A') :- $p2$(a,b), $p3$(b,'A',e), $p4$(a,f). □

Definition 3.2. (Filter containment) Given two conjunctive queries C_1 and C_2, C_1 *filter contains* C_2 iff there is a filter σ on C_2 such that C_1 contains $\sigma(C_2)$. The containment mapping from C_1 to σ (C_2) is called *filter mapping*. □

We regard constants in the body of the filter-contained query C_2 as distinguished variables of the query during filter containment test. Semantically, the filter containment of C_2 in C_1 tests whether a set of tuples contained in the result of C_1 can be obtained from the result of C_2. A filtered query (view) $\sigma(C_2)$ is a new query for obtaining the set of tuples. We consider only *optimal* filter σ on C_2. The optimal filter on C_2 represents the set of query conditions of C_1 that is not satisfied by C_2. Intuitively, a filtered view $\sigma(V)$ can be obtained from a given view V if V exports all the variables which correspond to the head variables of a partial query and the variables on which the set of query conditions of a partial query that is not satisfied in V is applied.

The following algorithm describes how to obtain a filter mapping from given two queries. In the algorithm, the procedure *new_predicate_maps*(P, V, H) tries to place a new set of predicate mappings from the body of P to the body of V set in H. There is no necessity for the predicate mappings in H to be consistent. If it fails to locate a new set of predicate mappings, it returns FALSE; otherwise, it returns TRUE. The procedure *applicable_filter*(H, f, σ, V) try to places the union of symbol mappings induced by H set in f and computes a *filter* σ associated with f obtained by the following rules: (1) For all $C \rightarrow V$ in f, where C is a constant and V a variable, V=C is added into σ, (2) For all V, if V is mapped to multiple variables $V_1, .., V_k$ by f, $V_i=V_1$, $1 \leq i \leq k$, is added into σ. It returns TRUE if a variable does not map to two different constants, a constant doesn't map to different constant under f and all variables in σ are one of the distinguished variables of V; otherwise, it return FALSE. It should be obvious that the filter we obtained is the set of qeury conditions of the partial query that is not satisfied by the original view definition. Finally, the algorithm returns the tupe $(\sigma(v(\overline{Y})), f, H)$ where $\sigma(v(\overline{Y}))$ is the head of filtered view by applying the optimal filter σ, f is a filter mapping, and H a set of predicate mappings from which f is obtained, if the mapping can be obtained; otherwise, it returns NULL.

```
Procedure FilteredView (P, V)
/* P and V are conjunctive queries and $v(\overline{Y})$ is the head of V */
while(new_predicate_maps(P, V, H)) {
    1. if not(applicable_filter(H, f, $\sigma$, V)), break.
    2. For each $V \rightarrow V_i$ in f and $V_i=V_1$ in $\sigma$, $1 \leq i \leq k$,
       remove $V \rightarrow V_i$ from f, $2 \leq i \leq k$.
    3. For each $C \rightarrow V$ in f and V=C in $\sigma$, replace $C \rightarrow V$
       in f with $C \rightarrow C$, where C is a constant.
    4. if some distinguished variable of P is not mapped
       to one of variable of $\sigma(v(\overline{Y}))$ under f, break.
    5. return ($\sigma(v(\overline{Y}))$, f, H).
}
return NULL.
```

Figure 1: finding a filtered instance from given a partial query and a view.

As shown in the algorithm, a filter containment mapping is computed from the union of induced symbol mappings from the body of partial query to the body of view. The algorithm checks whether the associated filter is applicable, eliminates the source of inconsistence from the union of induced symbol mappings, and checks the head variables of the partial query can be mapped to only the head variables of the view.

Theorem 3.1. Given two queries P and V, if V has no predicate repetitions in its body, the algorithm **FilteredView** solves the following problem in polynomial time: *Decide if P filter contains V, and if yes, find a filtered view of V.* □

The filter containment problem has the conventional conjunctive query containment problem within it. So, this problem is also NP-Completeness. The main exponential factor is the number of sets of predicate mappings from the body of a partial query to the body of a view that is to be considered.

In the algorithm, after a set of predicate mappings is determined, the remainder of the algorithm is completed in polynomial time. In the case where a view has no duplicated predicates, the set of predicate mappings from the body of a partial query to the body of a view is unique, if the set exists. Because it is clear that determining the existence of a predicate mapping and computing the mapping may be accomplished in polynomial time, computing the set may also be done in polynomial time, exactly in $O(nml)$, where n represents the number of subgoals in the partial query, m the number of subgoals in the view, and l the maximal size of a subgoal in the partial query. Due to lack of space, we shall omit the proof for soundness of the algorithm.

A contained rewriting is constructed by combining a set of filtered views after variable renaming. Variable renaming is done by applying the inverse of the associated filter mapping to the head of the filtered view. The following theorem provides the necessary condition for the existence of a rewriting using a view.

Theorem 3.2. *Given a query Q and a view V, if a contained rewriting R using the view exists then there is a partial query that filter contains the view.* □

Outline of Proof. Suppose that there is a contained rewriting R using the view V. Then there must exist a set of predicate mappings H that result in a containment mapping from Q to $Exp(R)$. Let $v(\overline{Y})$ be an instance of V in the rewriting R. Then there is a subset of query subgoals G that is covered by $v(\overline{Y})$. Now consider the partial query P over G having $p(\overline{X})$ as its head where $\overline{X}$ is a tuple of the distinguished variables of Q in G and non-distinguished shared query variables that appears in G as well as in any other subgoals in Q. By showing that P is the partial query that filter contains V, we will prove our theorem. To show that the P filter contains V, we consider the filtered view $v(\overline{Y})$:- $Exp(v(\overline{Y}))$. Let M represent a set of predicate mappings from each subgoal of P to a subgoal of $Exp(v(\overline{Y}))$ in H. We know that each pair of predicate mappings in M is consistent because M is a subset of H. Now, the remainder of this proof is the head variable mapping. If all variables in $\overline{X}$ are mapped to the variables in $\overline{Y}$ under M, our proof is completed. First, consider the case of distinguished variables. If a distinguished query variable maps to non-distinguished variable of the filtered view under M, M cannot be a subset of H because under H a distinguished query variable must map to a unique distinguished variable. Second, consider the case of non-distinguished shared variables. Let us suppose that a non-distinguished shared variable map to non-distinguished variable in the filtered view. Then the join condition for the shared variable in the query cannot be satisfied in the contained rewriting thereby the rewriting is not contained in the query. So all head variables of P must be mapped to the head variables of the filtered query. □

4. Rewriting queries using a set of views

In the previous chapter, we show that the filter containment test is used to determine whether a view can replace a partial query in a rewriting. In this chapter we summarize the conditions that are sufficient for the construction of contained rewritings.

Before the construction of contained rewritings, we find all the necessary filtered views using the filter containment tests for each view to construct all possible contained rewritings. We say that a view *consumes* a partial query or the subset of query subgoals that determine the partial query if a filter mapping from the partial query to the view exists. A filtered view I of a view is *minimal* if there is no filtered view I' for the same view such that I' consumes a subset of the set of subgoals that is consumed by I. In our approach, for each view, we try to find all of the minimal filtered views. Intuitively, minimal filtered views are sufficient to compute all possible contained rewritings because a set of subgoals, which cannot be consumed by one minimal filtered view, can be consumed by multiple minimal filtered views if the union of consumed subgoals of the filtered views is the same as the set. The following property describes the characterizations of minimal filtered views such that we can reduce the search space of finding minimal filtered views using it.

Property 4.1. (1) A minimal filtered view consumes only a single subgoal, or a set of subgoals such that some non-distinguished shared query variables appear in only the set. (2) The consumed subgoals of the minimal filtered instances for a view are disjointed. □

A contained rewriting is constructed by combining a set of filtered views after variable renaming. With a filter and a filter mapping necessary for variable renaming, another type of information, called *query constraint*, needed to combine filtered views correctly. Given a view and a partial query, a query constraint C is a set of equality predicates on the query variables. We modify the procedure *applicable_filter*($\sigma(v(\overline{Y}))$, f, H) in the procedure FilteredView(P,V) as *applicable_filter*($\sigma(v(\overline{Y}))$, f, H, C) to consider the query constrained C. It can be obtained by the following rules: (1) For all V→C in f, where C is a constant and V a variable, V=C is added into C, (2) If multiple variables V_1, .., V_k are mapped to the same variable V by f, V_i=V_1, $1 \le i \le k$, is added into C. When query variables map to constants or multiple query variables map to the same view variable under f, new equality predicates are introduced. This constraint must be observed in combining view instances.

Before combining filtered views, we annotate each filtered view with associated information. Given a filtered view, the annotated filtered view(AFV) is described as a tuple of the form ($\sigma(v(\overline{Y}))$, f, H, C, G) where $\sigma(v(\overline{Y}))$ is the head of the filtered view, f the associated filtered mapping, H a set of predicate mappings from which f is obtained, C the associated query constraint, and G the set of consumed subgoals.

```
Procedure findAFVs($Q$, $\Lambda$ )
/* Q is a query and $\Lambda$ is a set of views */
AFVs = {}.
For each view $V$ in $\Lambda$
    For each subgoal $g$ in $Q$
        Add to AFVs any new AFV ($\sigma(v(\overline{Y}))$, $f$, $H$, $C$, $G$) that can be constructed by the procedure
        FilteredView(G, V) where $\sigma(v(\overline{Y}))$ is the minimal filtered view of $V$ such that $G$ contains $g$ and the
        filter mapping $f$ exists.
Return AFVs.
```

Figure 2: Finding all AFVs

The algorithm for finding all necessary AFVs is shown in Figure 2. In our algorithm, first, we try to obtain a AFV for a set of single query subgoal. If the filter containment test for the set is fail, then we extend the set by adding shared variable subgoals, that is, the subgoals having shared variables which are in the failed subgoal and test the filter containment for the set again. We do this process until the set is unchanged or a AFV is obtained. Property4.1 is used during this process not to extend the testing set for unnecessary subgoal.

Consider the application of the algorithm to our example 2.1. The AFVs that will be created are shown in Figure 3. For V_1, we first consider a set {p1(x,y)} in the query. But we cannot find a filter mapping for that. Next, we extend the set by adding the shared variable subgoals p2(y,z). Now, we can find the result filter mapping {p1(x,y)→p1(a,b), p2(y,z)→p2(b,c)} so that V_1 consumes {p1,p2}. For p2(y,z), we need not to test filter containment, because p2 is already covered by V_1. For p3(z,'A',w), no filter mapping exists. The numbers in f means the subgoals of a query and views in that order. The numbers in G means the query subgoals in that order.

$\sigma(v(\overline{Y}))$	f	H	C	G
v_1(a1,c1,d1)	{x→a1, y→b1, z→c1}	{1→1, 2→2}	{}	{1, 2}
v_2(a2,'A',c2)	{z→a2, 'A'→'A', w→c2}	{3→1}	{}	{3}
v_3(b3,c3,'A',e3)	{z→c3, 'A'→'A', w→e3}	{3→2}	{}	{3}

Figure 3: AFVs fromed as part of our example

The following theorem summarizes the conditions that are sufficient for the construction of contained rewritings.

Theorem 4.1. Given a query Q with $q(\overline{X})$ as a head and AFVs $A_i = (v_i(\overline{Y}_i), f_i, C_i, G_i)$, $i = 1, \ldots, l$, the rewriting $q(\overline{X})$:- $f_1^{-1}(\sigma_1(v_1(\overline{Y}_1)))$, …, $f_l^{-1}(\sigma_l(v_l(\overline{Y}_l)))$ is correct if

1. $G_1 \cup .. \cup G_l$ = Subgoals(Q), and
2. for every $i \neq j$, $G_i \cap G_j = \varnothing$, and
3. for every $i \neq j$, no conflict in C_i and C_j. □

The proof is based on the theory of containment mapping. Due to lack of space, we shall omit the proof of the theorem in this paper. We use the inverse mapping of filter mapping for variable renaming of each filtered view. The fact that we only need to consider sets of filtered views that provide partitions of the query subgoals drastically reduces the search space of the algorithm.

The rewritings obtained after combining step still may contain redundant views. However, removing them involves the generation of new filtered views. Several filtered views can be replaced with a new filtered view. The following property states which filtered views can be replaced with one new filtered view:

Property 4.2. In a rewriting, the filtered views $I_1, \ldots, I_k$ that are generated from same view can be replaced with a new filtered view I if the corresponding set of predicate mappings $H_1, \ldots, H_k$ have view subgoals as targets that are pairwise disjointed in the view. □

New filtered view can be obtained by using filter mapping from the union of covered subgoals of the filtered views to the union of view subgoals which were targets of previous filter mappings.

5. Related works

Algorithms for rewriting query using views have been recently used to satisfy the various needs of several information integration systems [4, 7, 8]. Equivalent rewriting has been studied for use in query optimization [1, 14]. The approaches for contained rewriting are the bucket algorithm [7], using query folding [10], the inverse-rules algorithm [3], the shared bucket algorithm, and the Minicon algorithm [9]. The bucket algorithm is expensive because exponential containment test is needed to test the soundness of the rewritings. The inverse-rules algorithm works for recursive queries but the second stage of the algorithm where it puts together the inverse rules is almost as expensive as the bucket algorithm's exponential containment test. The shared bucket algorithm and the Minicon algorithm are similar to our algorithm in the approach that before constructing candidate rewritings, the role of a view in some rewriting is calculated to avoid the exponential containment test. Their solutions, however, were based on finding syntactic from the query variables to the view variables while we showed that the filter containment is the theoretical background for the contained rewriting problem. The filter containment is testing whether the query, whose result is contained in a partial query, can be obtained from a given view. Then the filtered instance, which was calculated during the test, is the query. Rewriting queries using views with specified binding patterns is considered in [7,8,11].

6. Conclusion

In this paper, we presented a solution for finding the maximally contained rewriting using views that introduces the concept of filter containment from the partial query of the query to a view and use it to generate only sound candidates, thereby eliminating the need for an exponential containment test from the query to the candidate. We believe that the filter containment supplies a firm theoretical foundation for the contained rewriting problem. We showed that, in a common case when predicate repetitions are not allowed in the bodies of views, the filter containment test is performed in polynomial-time.

Reference

[1] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, and K. Shim, "Optimizing queries with materialized views", In ICDE, 1995.

[2] A. Chandra and P. Merlin. Optimal implementation of conjunctive queries in relational databases. In Proceedings of the Ninth Annual ACM Symposium on Theory of Computing, pp. 77-90, 1977.

[3] O. M. Duschka and M. R. Genesereth, "Answering recursive queries using views", In Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Conference on Principles of Database Systems, PODS, May 1997.

[4] O. M. Duschka and M. R. Genesereth, "Query planning in infomaster", In Proceedings of the ACM Symposium on Applied Computing, 1997.

[5] Alon Y. Levy, Answering queries using views: A survey. 1999. Manuscript available from http://www.cs.washington.edu/homes/alon/views.ps.

[6] A. Y. Levy, A. O. Mendelzon, Y. Sagiv, and D. Srivastava, "Answering queries using views", In PODS, 1995.

[7] A. Y. Levy, A. Rajaraman, and J. J. Ordille, "Querying heterogeneous information sources using source descriptions", In VLDB, 1996.

[8] Y. Papakonstantinou, A. Gupta, and L. Haas, "Capabilities-based query rewriting in mediator systems", In IEEE Int. Conf. on Parallel and Distributed Information Systems, 1996.

[9] R. Plttinger and A. Y. Levy, "A Scalable algorithm for answering queries using views", In VLDB, 2000.

[10]X. Qian, "Query folding", In ICDE, pp 48-55, New Orleans, LA, 1996.

[11]A. Rajaraman, Y. Sagiv, and J. D. Ullman, "Answering queries using templates with binding patterns", In Proceeding of the Fourteenth ACM Symposium on Principles of Database Systems, pp. 105-112, 1995.

[12]J. D. Ullman, Principles of Database and Knowledge-Base Systems, Vols. I and II. Computer Science Press, New York, 1988.

[13]J. D. Ullman, "Information integration using logical views", In Proc. ICDT, Delphi, Greece, pp. 19-40.

[14]H.Z. Yang and P.-A. Larson, "Query Transformation for PSJ-queries", the 13th VLDB Conf. 1987.