Rewriting queries using views over monadic database schemas

Jan Van den Bussche
Limburg University, Belgium

1 Introduction

In research in databases and AI, the problem of rewriting queries using views has received a lot of interest recently; we refer to the nice survey by Levy for background [Lev99]. In this short note, we offer two remarks on the computational complexity of this problem.

First, we offer a most simple proof that the problem is NP-complete. We should specify immediately that we work with the *contained rewriting* variant of the notion of rewriting queries using views. For the *equivalent rewriting* variant, an NP-completeness proof was already published by Levy et al. [LMSS95].

Second, we restrict attention to database schemas with unary relations only. Such "monadic" database schemas have classically provided a special case where difficult problems become easy. For example, satisfiability of relational calculus sentences in general is undecidable, but is decidable if the database schema is monadic [BGG97]. We show that over monadic database schemas, the problem of rewriting queries using views is efficiently solvable.

2 Preliminaries

We begin by recalling the definitions of the needed notions. See the references [AHV95, Lev99, Ull89] for more background.

A *database schema* $\mathcal{S}$ is a finite set of relation names, where each relation name has an associated natural number, called its *arity*.

An *atomic formula* over $\mathcal{S}$ is an expression of the form $R(\bar{x})$, where $R \in \mathcal{S}$, and $\bar{x}$ is a k-tuple of variables, where k is the arity of R. A *conjunctive query*, which we will call just *query* for short, over $\mathcal{S}$, is an expression of the form $head \leftarrow body$, where *head* is a tuple of variables, and *body* is a finite set of atomic formulas over $\mathcal{S}$. These atomic formulas are called the *subgoals* of the query. The width of *head* determines the arity of the query.

Given two queries Q_1 and Q_2, we say that Q_1 is *contained* in Q_2 if there is a mapping f from the variables in Q_2 to those in Q_1, such that $f(head_2) = head_1$ and $f(body_2) \subseteq body_1$. Such a mapping is called a *homomorphism* from Q_2 to Q_1.

A *view schema* $\mathcal{V}$ over $\mathcal{S}$ is a finite set of relation names, disjoint from $\mathcal{S}$, where each relation name has an associated query over $\mathcal{S}$ (called a *view*).

We can look at view schemas as ordinary database schemas by ignoring the queries associated to the relation names. We can thus consider queries over view schemas. Let P be such a query over $\mathcal{V}$, and consider a subgoal $V(\bar{x})$ of P. By an *expansion* of this subgoal, we mean the body of V, in which we rename the head variables of V by the corresponding variables in $\bar{x}$, and rename the other variables to new, fresh variables. The *expansion* of P then is a query E over $\mathcal{S}$ obtained as follows. The head of E equals that of P. The body of E is formed by including an expansion of each subgoal of P.

We are finally ready to define the notion of rewriting queries using views. Let $\mathcal{S}$ be a database schema, Q a query over $\mathcal{S}$, and $\mathcal{V}$ a view schema over $\mathcal{S}$. Then Q can be *rewritten using* $\mathcal{V}$ if there exists a query P over $\mathcal{V}$ such that the expansion of P is contained in Q. We call P a *rewriting*. A rewriting P, with expansion E, is called *optimal* if for any other rewriting P', with expansion E', if E is contained in E', then E' is also contained in E (i.e., E and E' are equivalent).

3 NP-completeness

Lemma 1. *Let Q be nullary (i.e., with an empty head), and let $\mathcal{V}$ consist of a single view V which is also nullary. Then Q can be rewritten using $\{V\}$ if and only if V is contained in Q.*

Proof. Clearly, if V is contained in Q, then Q can be rewritten using $\{V\}$ because $() \leftarrow \{V()\}$ is a rewriting. Conversely, if Q can be rewritten using $\{V\}$, the only possible rewriting is $() \leftarrow \{V()\}$ (unless Q's body is empty, in

which case V is trivially contained in Q). The expansion of this rewriting, which is V, must be contained in Q. □

Theorem 1. *For any fixed database schema $\mathcal{S}$ having at least one binary relation name R, the following decision problem is NP-complete: Given Q and $\mathcal{V}$, can Q be rewritten using $\mathcal{V}$?*

Proof. The problem is well known to be in NP [LMSS95]. To show completeness, we reduce graph 3-colorability to it. Given a graph G, we construct a query Q and a single view V as follows. The body of Q equals $\{R(x, y) \mid (x, y)$ is an edge in $G\}$. (Note that the nodes of G play the role of variables.) The head of Q is empty. The body of V equals $\{R(r, g), R(g, r), R(r, b), R(b, r), R(g, b), R(b, g)\}$. Here, the variables r, g, and b intuitively stand for the three colors red, green, and blue. The head of V is again empty.

Now suppose G is 3-colorable, by a coloring f from the nodes of G to the colors $\{r, g, b\}$. This f is a homomorphism from Q to V, and thus Q can be rewritten using $\{V\}$ by Lemma 1.

Conversely, suppose Q can be rewritten using $\{V\}$, i.e., there is a homomorphism f from Q to V (Lemma 1). This homomorphism clearly gives us a 3-coloring of G. □

4 Monadic database schemas

If the arity of every $R \in \mathcal{S}$ is one, we call $\mathcal{S}$ *monadic*. Note that we needed a binary R in the proof of Theorem 1. In this section, we will show:

Theorem 2. *If $\mathcal{S}$ is monadic, the following problem is solvable in polynomial time: Given Q and $\mathcal{V}$, decide if Q can be rewritten using $\mathcal{V}$, and if yes, output an optimal rewriting.*

From now on we implicitly assume that the schema $\mathcal{S}$ is monadic. We first introduce some terminology: we call a subgoal $R(x)$ of any query *primary* if x occurs in the head; otherwise we call the subgoal *secondary*. When $R(x)$ is a secondary subgoal, the particular variable x is irrelevant, and from now on we identify such a secondary subgoal with just the relation name R.

We begin the proof of Theorem 2 with the following:

Lemma 2. *Q can be rewritten using $\mathcal{V}$ if and only if the following holds:*

1. *For every primary R-subgoal of Q, some $V \in \mathcal{V}$ also has a primary R-subgoal, for all relation names R;*

2. *For every secondary subgoal R of Q, some $V \in \mathcal{V}$ also has an R-subgoal (primary or secondary).*

Proof. Suppose Q can be rewritten using $\mathcal{V}$. Let P be a rewriting, and let E be its expansion. We have a homomorphism f from Q to E. Take a primary subgoal $R(x)$ of Q. Because f is a homomorphism, $R(f(x))$ is a primary subgoal of E. Hence, it must come from the expansion of a subgoal $V(\ldots, f(x), \ldots)$ of P. This V has a $R(f(x))$ as primary subgoal, as desired. The case of a secondary subgoal of Q is similar.

Conversely, suppose every subgoal of Q is "covered" in the sense of the lemma. We construct a rewriting P as follows. The head of P equals the head of Q. The body of P is formed as follows. For every primary subgoal $R(x)$ of Q, take a view V with a primary subgoal $R(y)$. In the head of V, rename all variables except y to fresh, new ones, and rename y to x; this gives us a tuple $\bar{z}$. Now include in the body of P the subgoal $V(\bar{z})$. Similarly, for every secondary subgoal R of Q, take a view V with an R-subgoal. This time, again in the head of V, rename all variables to new ones, yielding $\bar{z}$, and include the subgoal $V(\bar{z})$ in the body of P. By construction, there is a homomorphism from Q to the expansion of P, namely, the identity. □

Testing the conditions of the above lemma is very simple, and if they hold, the above proof gives us an $O(n^2)$ time algorithm to construct a rewriting. So, Lemma 2 gives a polynomial-time solution to the problem to decide whether Q can be rewritten using $\mathcal{V}$, and if so, to construct an arbitrary rewriting. To prove Theorem 2, however, we want a rewriting that is optimal. To this end, we need to refine our procedure.

Let P be a rewriting, and let E be its expansion. A secondary subgoal R of E is called *superfluous* if E also has a primary R-subgoal. We note:

Lemma 3. *A rewriting P (with expansion E, and with the same head as Q) is optimal if and only if for every other rewriting P' (with expansion E', and also with the same head as Q) such that E is contained in E', we have:*

1. *Every primary subgoal of E is also a primary subgoal of E';*

2. *Every non-superfluous secondary subgoal of E is also a secondary subgoal of E'.*

Proof. The if-direction is immediate, because the two conditions imply the existence of a homomorphism from E to E'. For the only-if direction, since P is optimal and E is contained in E', there is a homomorphism from E to E', which immediately yields (1) because their heads are the same. The containment of E in E' together with (1) implies that E and E' have exactly the same primary subgoals. Regarding (2), by the homomorphism, for any secondary subgoal R of E there is an R-subgoal of E'. If R is in addition non-superfluous in E, there is no primary R-subgoal in E, and thus not in E', whence R must be secondary in E'. □

Take an arbitrary rewriting P (with expansion E) produced by our algorithm. The head of P equals that of Q. Hence, by the above lemma, P is *not* optimal if and only if there exists another rewriting P' (with expansion E') with the following properties:

1. E is contained in E'. Since E' is contained in Q, and the head of E equals that of Q, we may assume without loss of generality that the head of E' also equals that of Q. Then, the containment of E in E' means that

   (a) every primary subgoal of E' is also a primary subgoal of E; and

   (b) for every secondary subgoal R of E', there is also an R-subgoal (primary or secondary) in E.

2. *If* the primary subgoals of E' are *exactly* those of E, *then* there is non-superfluous secondary subgoal of E that is not a secondary subgoal of E'.

This yields the following algorithm to go from the given rewriting P to an optimal one:

1. Pick an arbitrary primary subgoal of E that is not a primary subgoal of Q; denote it by α.

2. Try to find a rewriting P' by a refinement of the algorithm from the proof of Lemma 2. The refinement is that we only take views V satisfying the following:

   - In the case of a primary subgoal $R(x)$ of Q, where we must have a primary subgoal $R(y)$ in V:

- for all other subgoals $S(y)$ of V with that variable y, the subgoal $S(x)$ must be in E, *but it must not be* α;
- for all remaining T-subgoals of V, there must also be a T-subgoal in E, for all relation names T.

- In the case of a secondary subgoal R of Q, we simply require that for all T-subgoals of V, there must also be T-subgoal in E, for all relation names T.

3. There are two possibilities:

   - Step 2 was successful. So, we have found a rewriting P', with expansion E', such that E is contained in E', and such that primary subgoal α of E is *not* a primary subgoal of E'. By the above, this means that our current rewriting P is not optimal. Replace P by P', and repeat step 2 with another α.
   - We got stuck in step 2 because we could not find a suitable V at some point. This means that α must be a primary subgoal in the expansion of any rewriting. Repeat step 2 with another α.

   If we have exhausted all α's, we now know of our current P (with expansion E) that the expansion E' of any other rewriting P' such that E is contained in E', will have exactly the same primary subgoals as E.

4. Pick an arbitrary non-superfluous secondary subgoal of E; denote it by U.

5. Try to find a rewriting P' by a refinement of the algorithm from the proof of Lemma 2. The refinement is that we only take views V satisfying the following:

   - In the case of a primary subgoal $R(x)$ of Q, where we must have a primary subgoal $R(y)$ in V:
     - for all other subgoals $S(y)$ of V with that variable y, the subgoal $S(x)$ must be in E;
     - there is no U-subgoal in V;
     - for all remaining T-subgoals of V, there must also be a T-subgoal in E, for all relation names T.

- In the case of a secondary subgoal R of Q, we simply require that for all T-subgoals of V, there must also be T-subgoal in E, for all relation names T, and additionally that there is no U-subgoal in V.

6. Again there are two possibilities:

   - Step 5 was successful. So, we have found a rewriting P', with expansion E', such that E is contained in E' and have exactly the same primary subgoals, while non-superfluous secondary subgoal U of E is *not* a secondary subgoal of E'. By the above, this means that our current rewriting P is not optimal. Replace P by P', and repeat step 5 with another U.
   - We got stuck in step 5 because we could not find a suitable V at some point. This means that U must be a secondary subgoal in the expansion of any rewriting. Repeat step 2 with another U.

   After we have exhausted all U's, our current P is optimal, as desired.

The above algorithm has $O(n^5)$ time complexity. Indeed, recall that the basic algorithm has $O(n^2)$ complexity. The refinement used in step 2 can be implemented in $O(n^4)$ time (extra checks for the presence of subgoals in E need to be done, and E can be $O(n^2)$ large), while the refinement used in step 5 can be implemented in the same $O(n^2)$ time. Both these refinements are run $O(n)$ times, giving $O(n^5)$. (Using suitable data structures to represent conjunctive queries it will surely be possible to improve this rough upper bound.) Theorem 2 is thereby proven.

5 Concluding remark

As mentioned in the Introduction, the *equivalent rewriting* variant of the problem of rewriting queries using views requires of a rewriting not only that its expansion is contained in Q, but that it is actually equivalent to Q. If an equivalent rewriting exists, it is clearly also an optimal contained rewriting, and conversely, every optimal contained rewriting will in fact be an equivalent rewriting. Hence, our algorithm for finding an optimal contained rewriting can be used to find an equivalent rewriting as well: it suffices to check at the end that Q is contained in the expansion of the found rewriting. This check can be implemented in polynomial time over monadic schemas.

References

[AHV95] S. Abiteboul, R. Hull, and V. Vianu. *Foundations of Databases*. Addison-Wesley, 1995.

[BGG97] E. Börger, E. Grädel, and Y. Gurevich. *The Classical Decision Problem*. Springer, 1997.

[Lev99] A.Y. Levy. Answering queries using views: A survey. Computer Science Department, University of Washington, 1999.

[LMSS95] A.Y. Levy, A.O. Mendelzon, Y. Sagiv, and D. Srivastava. Anwering queries using views. In *Proceedings 14th ACM Symposium on Principles of Database Systems*, pages 95–104. ACM Press, 1995.

[Ull89] J.D. Ullman. *Principles of Database and Knowledge-Base Systems*, volume II. Computer Science Press, 1989.