Approximate Rewriting of Queries Using Views

Foto Afrati[1], Manik Chandrachud[2], Rada Chirkova[2], and Prasenjit Mitra[3]

[1] National Technical University of Athens, afrati@softlab.ntua.gr
[2] Computer Science Department, NC State University, Raleigh, NC USA
mnchandr@ncsu.edu, chirkova@csc.ncsu.edu
[3] College of Information Sciences and Technology, Pennsylvania State University, University Park, PA USA, pmitra@ist.psu.edu

Abstract. We study *approximate,* that is contained and containing, rewritings of queries using views. We consider conjunctive queries with arithmetic comparisons (CQACs), which capture the full expressive power of SQL select-project-join queries. For contained rewritings, we present a sound and complete algorithm for constructing, for CQAC queries and views, a maximally-contained rewriting (MCR) whose all CQAC disjuncts have up to a predetermined number of view literals. For containing rewritings, we present a sound and efficient algorithm *pruned-MiCR,* which computes a CQAC containing rewriting that does not contain any other CQAC containing rewriting (i.e., computes a minimally containing rewriting, MiCR) *and* that has the minimum possible number of relational subgoals. As a result, the MiCR rewriting produced by our algorithm may be very efficient to execute. Both algorithms have good scalability and perform well in many practical cases, due to their extensive pruning of the search space, see [1].

1 Introduction

Rewriting queries using views and then executing the rewritings to answer the queries is an important technique used in data warehousing, information integration, query optimization, and other applications, see [2, 3, 4, 5, 6] and references therein. A large amount of work has been done on obtaining *equivalent rewritings* of queries, that is, rewritings that can be used to derive *exact* query answers (see, e.g., [7, 8, 9]). When equivalent rewritings cannot be found, then in many applications it makes sense to work with *contained rewritings,* which return a subset of the set of the query answers. Of special interest in this context are *maximally contained rewritings (MCRs),* which can be used to obtain a maximal subset of the query answers that can be obtained using the given views (see, e.g., [10, 4, 11, 12, 13]). In addition, in applications such as querying the World-Wide Web, mass marketing, searching for clues related to terrorism suspects, or peer data-management systems (see, e.g., [14, 15]), users prefer to get a *superset* of the query answers, rather than getting no answers at all (when no equivalent or contained rewritings exist). In such scenarios, users might be interested in *containing rewritings,* which return a superset of the set of the query

answers. *Minimally containing rewritings (MiCRs)* [16, 17, 18] are the containing rewritings that return the fewest false positives when answering the query.

In this paper we study maximally contained and minimally containing rewritings of queries using views, which we refer to collectively as *approximate rewritings.* We focus on conjunctive queries with arithmetic comparisons *(CQACs),* that is on the language capturing the full expressive power of practically important SQL select-project-join (SPJ) queries. (The well-understood language of conjunctive queries [19] does not capture the in- or non-equalities that are characteristic of SQL SPJ queries.) Specifically, we assume CQAC queries and views, and consider CQAC rewritings, possibly with unions *(UCQACs).* The well-studied (for conjunctive queries and views) problems of finding equivalent rewritings and MCRs are recognized as being significantly more complex for CQACs, with many practically important cases still unexplored [4, 13]. The complexity of the problems in the presence of ACs is mainly due to the more complex containment test — the containment test is NP-complete in the case of CQs [19] but Π_2^P-complete [4, 20] in the case of CQACs. We illustrate the challenge by an example from [12].

Example 1. Consider CQAC query Q and CQAC view V, both defined using binary predicate p, as well as a CQAC query R defined in terms of the view V. Let $Q()$:- $p(A, B),\ A \leq B$; $V()$:- $p(X, Y),\ p(Y, X)$; and $R()$:- $V()$. Here, R is a contained rewriting of Q; this containment can be verified using the containment tests of [4, 20] (see Sect. 2). Observe that the containment cannot be established using a single containment mapping [19] from Q to the expansion of R. □

Some of the authors of this paper presented in [9] a sound and complete algorithm that returns a UCQAC *equivalent* rewriting of the input CQAC query in terms of the input CQAC views. In this paper we focus on those problem settings where one is to find a rewriting of a given CQAC query in terms of given CQAC views, but an equivalent UCQAC rewriting does not exist, and thus the algorithm of [9] returns no answer. Further, Deutsch, Ludaescher, and Nash [16] provided approaches for solving the problem of rewriting queries using views with limited access patterns under integrity constraints, focusing on queries, views, and constraints over unions of conjunctive queries with negation. We comment on the contributions of [16] w.r.t. the algorithms that we propose in this paper when discussing our specific contributions.

The specific contributions presented in this paper are as follows:

1. **Contained rewritings:** Pottinger and Halevy developed algorithm MiniCon IP [12], which efficiently finds UCQAC MCRs for special cases of CQAC queries, views, and rewritings, specifically for those cases where the "homomorphism property" [21, 22] holds between the expansions of the rewritings and the query.[4] At the same time, MiniCon IP cannot find the rewriting R for the problem input of Example 1. We present a sound and complete algorithm called *Build-MaxCR,* for constructing a UCQAC size-limited MCR

[4] The *homomorphism property* is said to hold between CQAC queries Q_1 and Q_2 when a single mapping from Q_1 to Q_2 establishes the containment of Q_2 in Q_1; see Sect. 2.

(that is, an MCR that has up to a predetermined number of view literals) of arbitrary CQAC queries using arbitrary CQAC views.[5] The size-limit restriction of Build-MaxCR is due to the fact that for CQAC queries and views, a view-based UCQAC MCR may have an unbounded number of CQAC disjuncts, see Example 2. To the best of our knowledge, the approaches of [16] do not provide for constructing size-limited contained rewritings of the input queries using views, which are addressed by our algorithm Build-MaxCR.

2. **Containing rewritings:** We focus on the problem of enabling a MiCR of a CQAC query using CQAC views to be executed as efficiently as possible. To that end, we look at minimizing the number of relational subgoals of a given MiCR. Our main contribution is a sound and efficient algorithm that we call *pruned-MiCR*. Given a CQAC MiCR for a given problem input (CQAC query and views), pruned-MiCR performs *global* minimization of the MiCR, and in many cases produces MiCR formulations whose evaluation costs are significantly lower than those of the (MiCR) input to the algorithm. To the best of our knowledge, other approaches for MiCRs [16, 17, 18] do not involve minimization of the number of relational subgoals of the MiCRs.
3. **Reducing runtime of containment checking:** Finally, we study the problem of reducing the runtime of containment checking between two CQAC queries, and propose a runtime-reduction technique that takes advantage of some attributes drawing values from disjoint domains. (Intuitively, it does not make sense to compare the values of, e.g., attributes "price" and "name".) This technique can be used in a variety of algorithms. Specifically, it is applicable to our proposed algorithms Build-MaxCR and pruned-MiCR. Due to the space limit, this result (as well as our NP-completeness result for the problem of determining whether a CQAC containing rewriting exists for a given CQAC problem input, see Table 1) is omitted from this paper but can be found in the full version [1] of our paper, available online.

Table 1 gives a summary of our results and contributions.

Table 1. Our contributions, previous work, and applications.

	Contained Rewritings	**Containing Rewritings**
Decidability	UCQAC size-limited MCR for CQACs	UCQACs with negation [16]
Complexity	CQ: NP [7]	CQAC homomorphism property: NP-complete [1]
Algorithms	Size-limited UCQAC MCRs for CQACs	Global minimization of MiCR for CQACs
Previous Work	MCR [10, 13]	MiCR [16, 17, 18]
Applications	Data warehousing, security, privacy	Mass marketing, P2P, information retrieval

[5] Specifically, Build-MaxCR can find the rewriting R of Example 1.

Due to the space limit, we present here only a foundational exposition of our algorithms. The full version [1] of this paper provides all the details as well as our experimental results. While the running-time complexity of our proposed algorithms is high in the worst case (doubly exponential for algorithm Build-MaxCR, and singly exponential for algorithm pruned-MiCR), our experimental results indicate that both algorithms have good scalability and perform well in many practical cases, due to their extensive pruning of the search space.

Related Work

The problem of using views in query answering [7] is relevant in applications in information integration [4], data warehousing [10], web-site design [23], and query optimization [6, 7, 24]. Algorithms for finding rewritings of queries using views include the bucket algorithm [17, 25], the inverse-rule algorithm [26, 27, 28], the MiniCon algorithm [12], and the shared-variable-bucket algorithm [11]; see [10] for a survey. Almost all of the above work focuses on investigating MCRs or equivalent rewritings [4, 8], as it takes its motivation mostly from information integration and query optimization. Query-rewriting algorithms depend upon efficient algorithms for checking query containment. Existing work on query containment show that adding arithmetic comparisons to queries and views makes these problems significantly more challenging [29, 21, 20].

Since we consider rewritings that may return false positives or false negatives, our work has similarities with approximate answering of queries using views, see [30, 31, 32, 33] and references therein, as well as a detailed discussion in [1]. Approximate query answering is useful when exact query answers cannot be found, and the user would rather have a good-quality approximate answer returned by the system. Our approaches provide such approximate answers in the form of maximally contained or containing rewritings.

The problem of finding containing rewritings of queries using views has been studied in [17] in [16, 18]. Please see the beginning of Sect. 1 for a detailed discussion of the work of [16]. Other related work includes the results of Rizvi et al. [34], where query-rewriting techniques are used for access control, and the work of Miklau et al. [35], which contains a formal probabilistic analysis of information disclosure in data exchange under the assumption of independence among the relations and data in a database. Related work in security and privacy includes [36]. Calvanese et al. [37] discussed query answering, rewriting, and losslessness with respect to two-way regular path queries. In our work, we concentrate only on query rewritings.

2 Preliminaries

In this section we review some standard concepts related to answering queries using views, and introduce some notation that we will use throughout the paper.

2.1 Queries, Containment, and Views

We consider *conjunctive queries with arithmetic comparisons* (CQACs), that is, SQL select-project-join queries with equality and arithmetic-comparison selection conditions. Each *arithmetic comparison* (AC) subgoal is of the form $X\ \theta\ Y$ or $X\ \theta\ c$,[6] where the comparison operator θ is one of $<$, $\leq$, $>$, $\geq$, and $\neq$. We assume that database instances are over densely totally ordered domains. A variable is called *distinguished* if it appears in the query head. In the rest of the paper, for a query Q we denote the conjunction of all relational subgoals in Q as Q_0 and the conjunction of all ACs in Q as β. All the queries we consider are *safe,* that is each distinguished variable or variable appearing in the β of the query also appears in at least one relational subgoal of the query.

Definition 1. (Query containment) *A query Q_1 is* contained *in a query Q_2, denoted $Q_1 \sqsubseteq Q_2$, if and only if, for all databases D, the answer $Q_1(D)$ to Q_1 on D is a subset of the answer $Q_2(D)$ to Q_2 on D, that is, $Q_1(D) \subseteq Q_2(D)$.*

Chandra and Merlin [19] have shown that a CQ Q_1 is contained in another CQ Q_2 of the same (head) arity if and only if there exists a *containment mapping* from Q_2 to Q_1. The containment mapping is a (body) *homomorphism* h from the variables of Q_2 to the variables and constants of Q_1 and from the constants of Q_2 to themselves, that is for each subgoal $p(Z_1, \ldots, Z_n)$ of Q_2 it holds that $p(h(Z_1), \ldots, h(Z_n))$ is a subgoal of Q_1. In addition, for h to be a containment mapping from Q_2 to Q_1, it must be that the list $(X_1, \ldots, X_k)$ of the variables and constants in the head of Q_1 be $(h(Y_1), \ldots, h(Y_k))$ (that is, $X_i = h(Y_i)$ for $\forall i \in \{1, \ldots, k\}$), where $Q_2(Y_1, \ldots, Y_k)$ is the head of Q_2.

The containment test for CQACs is more involved. There are two ways to test the containment of CQAC Q_1 in CQAC Q_2 [29, 21]. We describe them briefly here; for more details see, e.g., [38]. The first test uses the notion of a *canonical database*: For each relational subgoal $p_i(\bar{X}_i)$ in Q, a canonical database for Q contains one tuple t in the base relation p_i, such that t is the list of "frozen" variables and constants from $\bar{X}_i$ (i.e., in forming t each variable in $\bar{X}_i$ is "frozen" to a unique constant except that equated variables are frozen to the same constant and each constant in $\bar{X}_i$ is kept as it is). We define one canonical database for each total ordering of the variables and constants in Q_1 that satisfies the ACs in Q_1. The test says that Q_1 is contained in Q_2 if and only if Q_2 computes, on all the canonical databases of Q_1, all the head tuples of Q_1.

The second containment test, see Theorem 1, uses the notion of a normalized version of a CQAC query. An equivalent *normalized version* [21, 39] Q' of a CQAC query Q does not have constants or repetitions of variable names in relational subgoals and has compensating built-in equality conditions.

Theorem 1. *For CQAC queries Q_1 and Q_2, $Q_1 \sqsubseteq Q_2$ iff implication ϕ holds:*

$$\phi : \beta_1' \Rightarrow \mu_1(\beta_2') \vee \ldots \vee \mu_k(\beta_2')$$

[6] We use uppercase letters to denote variables and lowercase letters for constants.

where μ_i's are all the containment mappings from Q'_2 to Q'_1 and β'_i is a conjunction of all the ACs in Q'_i, $i \in \{1, 2\}$. That is, the ACs in the normalized version Q'_1 of Q_1 logically imply (denoted "$\Rightarrow$") the disjunction of the images of the ACs of the normalized version Q'_2 of Q_2 under each mapping μ_i.

If there exists a containment mapping μ_i such that the right-hand side of ϕ is reduced to only one $\mu_i(\beta'_2)$, then we say the *homomorphism property* holds between Q_1 and Q_2. Afrati et al. [38] showed that when the homomorphism property holds, the implication can be checked on the queries without normalizing them. Checking CQAC containment is less complex in that case, because we need to check for the existence of just one mapping that satisfies the implication.

2.2 Rewriting Queries using Views

We consider the problem of finding rewritings under the *closed-world assumption (CWA)* [8], where for a given database, each view instance stores exactly the tuples satisfying the view definition. In addition, we consider *contained rewritings* under the *open-world assumption (OWA)* [8, 25]. Here, the views are sound but not necessarily complete, that is a view instance might store only some of the tuples satisfying the view definition.

Suppose we are looking for an answer to query Q on database D, and our access to D is restricted to using a set of views $\mathcal{V} = \{V_1, \ldots, V_m\}$. So instead of directly evaluating Q on D, we rewrite Q in terms of $\mathcal{V}$ and then evaluate the rewriting on D. We consider the following types of rewritings R of Q using $\mathcal{V}$. Here, $D_{\mathcal{V}}$ is the result of adding to database D the answers to views $\mathcal{V}$ on D.

Definition 2. (Rewritings)

1. *a. (CWA) R is a* contained rewriting *of Q using $\mathcal{V}$ under the CWA iff $R(D_{\mathcal{V}}) \subseteq Q(D)$ for all databases D.*
   b. (OWA) R is a contained rewriting *of Q using $\mathcal{V}$ under the OWA iff $R(I_{\mathcal{V}}) \subseteq Q(D)$ for all databases D and view instances $I_{\mathcal{V}}$ such that $I_{\mathcal{V}} \subseteq D_{\mathcal{V}}$.*
2. *(CWA) R is a* containing rewriting *of Q using $\mathcal{V}$ iff $\forall\ D : Q(D) \subseteq R(D_{\mathcal{V}})$.*
3. *(CWA) R is an* equivalent rewriting *of Q using $\mathcal{V}$ iff $\forall\ D : Q(D) = R(D_{\mathcal{V}})$.*

Since the answer to a containing rewriting R on a database D must contain all tuples that occur in the answer to Q on D, containing rewritings make sense only when the views that are used in constructing the containing rewriting are complete. Hence, containing rewritings are considered only under the CWA and not under the OWA. The same is true for equivalent rewritings, since an equivalent rewriting of Q is a rewriting that is a contained as well as a containing rewriting of Q. At the same time, since the result of a contained rewriting is allowed to leave out some of the answers to Q, contained rewritings make sense under the CWA and under the OWA.

Given a query Q and a set of views $\mathcal{V}$, for deciding whether there exists a contained (or containing) rewriting of Q using $\mathcal{V}$, we need to know the language in which we are allowed to construct rewritings. In the rest of the paper we

assume, unless otherwise stated, that the language of the rewritings for the existence problem is UCQACs.

We define the expansion of a rewriting as follows:

Definition 3. (Expansion of rewriting) *For a CQAC rewriting R that is expressed in terms of CQAC views $\mathcal{V}$, an* expansion R^{exp} *of R is obtained by replacing each view subgoal in R by the all the subgoals in the definition of that view. Each existentially quantified variable in the definition of a view in R is replaced by a unique variable in R^{exp}. For a UCQAC rewriting, the expansion is the union of the expansions of the CQACs that occur in that UCQAC.*

The evaluation of contained rewritings cannot return false positives, the evaluation of containing rewritings cannot return false negatives, and the evaluation of equivalent rewritings cannot return either false positives or false negatives. We will use the term *rewriting* to mean a contained or a containing rewriting; we will specify the type whenever it is not obvious from the context.

Theorem 2 is based on Definitions 2 and 3 and gives the tests for determining whether a CQAC rewriting R is a contained (or containing) rewriting of a CQAC query Q using CQAC views $\mathcal{V}$.

Theorem 2. *Let $Q, V_1, \ldots, V_m$ be CQAC queries defined on database schema $\mathcal{D}$, and let R be a CQAC rewriting of Q using $\{V_1, \ldots, V_m\}$. Then*

1. *R is a contained rewriting of Q if and only if $R^{exp} \sqsubseteq Q$.*
2. *R is a containing rewriting of Q if and only if $Q \sqsubseteq R^{exp}$.*

3 Algorithm Build-MaxCR: Finding MCRs for CQACs

In this section we present a sound and complete algorithm *Build-MaxCR,* for constructing a UCQAC size-limited maximally-contained rewriting (i.e., MCR with up to a predetermined number of view literals) of CQAC queries using CQAC views. We discuss the pseudocode and formulate the correctness results for the algorithm. These results resolve in the positive the problem of decidability of the existence of a UCQAC size-limited MCR for CQAC queries and views.

3.1 The Setting and Definitions

Suppose we are given a CQAC query Q and a set $\mathcal{V}$ of CQAC views, such that each of R_1 and R_2 is a CQAC contained rewriting of Q using $\mathcal{V}$. It is easy to see that the union $R_1 \cup R_2$ is also a contained rewriting of Q using $\mathcal{V}$. This observation motivates us to consider the language of *unions* of CQAC queries for maximally contained rewritings of CQAC queries using CQAC views. Given a CQAC query Q and a set $\mathcal{V}$ of CQAC views, a UCQAC contained rewriting R of Q using $\mathcal{V}$ is a *maximally-contained rewriting (MCR)* of Q using $\mathcal{V}$ in the language of UCQACs if for each UCQAC contained rewriting R' of Q using $\mathcal{V}$ it holds that $(R')^{exp} \sqsubseteq R^{exp}$.

The first question we examine is whether such a UCQAC MCR is always bounded in size. Consider an example based on the ideas from [22].

Example 2. Let query Q and views V_1 and V_2 be defined as follows. Let $Q() :- p(X,Y),\ p(Y,Z),\ s(Y),\ X \geq 2,\ Z \leq 7$; let $V_1(L,M) :- p(L,M),\ L \geq 2,\ M \leq 7$; and let $V_2(A,C) :- p(A,B),\ p(B,C),\ s(A),\ s(C)$.

We can show that each of R_3 and R_4 is a CQAC contained rewriting of Q using V_1 and V_2. Here, $R_3() :- V_1(L_1,A_1),\ V_2(A_1,C_1),\ V_1(C_1,M_2)$; and $R_4() :- V_1(X,T_1),\ V_2(T_1,T_2),\ V_2(T_2,T_3),\ V_1(T_3,Z)$.

Further, one can use the template of R_3 and R_4 to build rewritings R_5 (which has one extra V_2 subgoal as compared to R_4), R_6 (two extra V_2 subgoals), and so on. (See [1] for the details.) In the family of rewritings $\mathbf{R} = \{R_3, R_4, R_5, R_6, \ldots,\}$ that we build in this manner, each rewriting R_i (for $i \geq 3$) has two properties:

- the expansion of R_i is contained in Q, and
- R_i (for $i > 3$) is not contained in R_j for any $3 \leq j < i$.

Therefore, a UCQAC maximally contained rewriting of Q in terms of $\{V_1, V_2\}$ must include every R_i in the *infinite-cardinality* family $\mathbf{R}$. □

The point of Example 2 is that the number of CQAC disjuncts (such as R_i's in the example) in the maximally-contained UCQAC rewriting of a CQAC query using CQAC views may not be bounded, provided that the language of rewritings is UCQAC. Hence an algorithm for finding the UCQAC-MCR may not terminate on some CQAC inputs.

To address this problem, we introduce the concept of size-limited MCRs. Specifically, we define the problem of constructing a UCQAC size-limited MCR for a CQAC query using CQAC views. We use the following definition:

Definition 4. (A k-bounded (CQAC, UCQAC) query) *Given a database schema $\mathcal{V}$ and a positive integer number k. (1) A CQAC query Q defined on $\mathcal{V}$ is a* k-bounded (CQAC) query using $\mathcal{V}$ *if, for the number n of relational subgoals of Q, we have $n \leq k$. (2) $Q = \bigcup_i Q_i$ is a* k-bounded UCQAC query using $\mathcal{V}$ *if each CQAC component Q_i of Q is a k-bounded query using $\mathcal{V}$.*

Now, the *problem of constructing a UCQAC size-limited (k-bounded) MCR for a CQAC query using CQAC views* is specified as follows:

1. The *problem input* is a triple $(Q, \mathcal{V}, k)$, where Q is a CQAC query, $\mathcal{V}$ is a finite set of CQAC views, and k is a natural number.
2. The *problem output* is a UCQAC query $P = \bigcup_j P'_j$ in terms of $\mathcal{V}$, such that:
   (a) P^{exp} is contained in Q, $P^{exp} \sqsubseteq Q$;
   (b) P is a k-bounded (UCQAC) query in terms of $\mathcal{V}$; and
   (c) for each k-bounded UCQAC query R in terms of $\mathcal{V}$ such that $R^{exp} \sqsubseteq Q$, we have that $R^{exp} \sqsubseteq P^{exp}$.

Our proposed algorithm Build-MaxCR solves the above problem for arbitrary inputs $(Q, \mathcal{V}, k)$ as defined in the problem formalization. Our soundness and completeness results for Build-MaxCR (Sect. 3.3) establish that for each such input $(Q, \mathcal{V}, k)$, Build-MaxCR returns a maximally contained rewriting of Q in the language of k-bounded UCQAC queries over $\mathcal{V}$, if such a rewriting exists.

3.2 Our Algorithm Build-MaxCR

We now discuss briefly our algorithm Build-MaxCR, please see [1] for the pseudocode and examples. The general idea of the algorithm is to do a complete enumeration of the CQ parts, call them $\bar{P}_j$, of k-bounded CQAC queries defined on schema $\mathcal{V}$. (For a CQAC query R, we use the term "CQ part of R" to refer to the join of all relational subgoals of R, taken together with all the equality ACs implied by R.) For each such $\bar{P}_j$, the algorithm associates with $\bar{P}_j$ a *minimum* set S'_j of inequality/nonequality ACs on the variables and constants of $\bar{P}_j$, such that S'_j ensures containment of $\bar{P}_j^{exp}\&S'_j$ in Q. The output for Build-MaxCR is the union P of all the CQAC queries $\bar{P}_j\&S'_j$ for which the containment holds. (By [21], $\bar{P}_j^{exp}\&S'_j \sqsubseteq Q$ for each j ensures $P^{exp} \sqsubseteq Q$, where $P = \bigcup_j \bar{P}_j\&S'_j$.)

The algorithm uses the notion of a "CQAC-rewriting template" for a problem input $(Q, \mathcal{V}, k)$. For an input of this form, Build-MaxCR enumerates all *cross products*, call them P_i, of up to k relational subgoals in terms of $\mathcal{V}$. We call each P_i, with $s \leq k$ subgoals, a *CQAC-rewriting template (for Q) of size s.*

Another notion used by the algorithm is that of a "MaxCR canonical database." Given query Q and its CQAC-rewriting template P (of some size s), the set $\mathcal{D}_P^Q$ of *MaxCR canonical databases for Q and P* is constructed in the same way as the set $\mathcal{D}^{(P^{exp})}$ of canonical databases of the expansion P^{exp} of P (see Sect. 2). The only difference is that the set W of constants and variables of P^{exp} (W is used in the construction of $\mathcal{D}^{(P^{exp})}$) is extended, for the construction of $\mathcal{D}_P^Q$, to include also all the numerical constants of the query Q.

3.3 Correctness of Algorithm Build-MaxCR

We now formulate theorems that establish soundness and completeness of Build-MaxCR, as well as the decidability results for two decision versions of the problem of constructing UCQAC k-bounded MCRs for CQAC queries using CQAC views. The proofs of these correctness results for Build-MaxCR, as well as our experimental results that corroborate the efficiency and scalability of the algorithm, can be found in the full version [1] of the paper.

Theorem 3. (Soundness of Build-MaxCR) *For a Build-MaxCR problem input $(Q, \mathcal{V}, k)$, let P be a CQAC-rewriting template (of some size $s \leq k$). Then for any CQAC query $P' :- P\&S$ that is output by Build-MaxCR, $(P')^{exp}$ is contained in Q. (Here, S is a conjunction of ACs.)*

(For the notion of a "CQAC-rewriting template" for a Build-MaxCR problem input $(Q, \mathcal{V}, k)$, please see Sect. 3.2.)

Theorem 4. (Completeness of Build-MaxCR) *For a Build-MaxCR problem input $(Q, \mathcal{V}, k)$, let $\mathbf{R}$ be a UCQAC query defined in terms of $\mathcal{V}$, such that (i) in each CQAC component R_i of $\mathbf{R}$, the number of relational subgoals of R_i does not exceed k, and (ii) $\mathbf{R}^{exp} \sqsubseteq Q$. Then (1) the output of Build-MaxCR is not empty, and (2) denoting by $\mathbf{P}$ the UCQAC output of Build-MaxCR, we have that $\mathbf{R}^{exp} \sqsubseteq \mathbf{P}^{exp}$.*

By Theorems 3 and 4 we obtain immediately the following two results:

Theorem 5. (Decidability) *Given a CQAC query Q, a set $\mathcal{V}$ of CQAC views, and a natural number k. (1) It is decidable to determine whether Q has a UCQAC k-bounded contained rewriting in terms of $\mathcal{V}$. (2) Further, given in addition a UCQAC k-bounded query R defined in terms of $\mathcal{V}$, the problem of determining whether R is a UCQAC k-bounded MCR for Q using $\mathcal{V}$ is decidable.*

4 Finding Minimized Minimally Containing Rewritings

We now turn to the problem of finding minimally containing rewritings [16, 17, 18], which we abbreviate as *MiCRs,* of a CQAC query using CQAC views. The word "minimal" in "MiCR" refers to a containing rewriting that contains the fewest false positives (in the given rewriting language) w. r. t. the query answer.

We focus on the problem of enabling a MiCR of a CQAC query using CQAC views that can be executed efficiently. To that end, we look at minimizing the number of relational subgoals of a given MiCR, and thus the number of joins in the evaluation plans for the MiCR. In Sect. 4.1, we introduce the notion of a *minimized MiCR.* The main contribution of this section is an algorithm that we call *pruned-MiCR,* see Sect. 4.2. Given a CQAC MiCR for a given problem input (i.e., for a CQAC query and a set of CQAC views), pruned-MiCR *globally* minimizes the MiCR in an *efficient* and *scalable* way. (See Sect. 4.3 for the correctness and complexity results.) Our experimental results [1] suggest that for many problem inputs (for the MiCRs for queries and views of certain types), pruned-MiCR outputs minimized MiCRs whose evaluation costs are significantly lower than those of the (MiCR) input to the algorithm.

Note that the idea of minimizing the number of subgoals in a rewriting is quite general and thus applicable beyond containing rewritings. Specifically, a straightforward modification of pruned-MiCR could be used to reduce the number of relational subgoals of (and thus to provide more efficient execution options for) the outputs of our Build-MaxCR algorithm of Sect. 3. See [1] for the details.

4.1 The Definitions

First, we provide a general definition of a MiCR and then we define (CQAC) minimized MiCRs.

Definition 5. (Minimally containing rewriting) *A query Q' defined in query language $\mathcal{L}_1$ is a* minimally containing rewriting (MiCR) *of a query Q defined in language $\mathcal{L}_2$ using a set of views $\mathcal{V}$ defined in language $\mathcal{L}_3$ if: (1) Q' is a containing rewriting of Q in terms of $\mathcal{V}$, and (2) there exists no containing rewriting (in language $\mathcal{L}_1$) Q'' of Q using $\mathcal{V}$, such that the expansion of Q'' is properly contained in the expansion of Q'.*

For the results in this section, each of $\mathcal{L}_1$ through $\mathcal{L}_3$ is the language of CQAC queries.

We now define the notion of *minimized MiCR.*

Definition 6. (Minimized MiCR) *Given a CQAC query Q and a set of CQAC views $\mathcal{V}$, CQAC MiCR R of Q using $\mathcal{V}$ is a* minimized (CQAC) MiCR *of Q using $\mathcal{V}$ if removing any relational subgoal of R results in query R' such that R and R' are not equivalent as expansions, that is $R^{exp} \not\equiv (R')^{exp}$.*

By definition, if we delete even a single relational subgoal from a minimized MiCR, it no longer remains a MiCR. Finding minimized MiCRs is especially important where the MiCR is computed once and then executed repeatedly. In such cases, it is important that the MiCR execute efficiently. Since a minimized MiCR may have fewer relational subgoals than the original MiCR (see, e.g., Example 3), and thus fewer joins, such a performance improvement would have a significant payoff.

We now introduce the notion of a "globally minimal" minimized MiCR. A *globally minimal minimized CQAC MiCR* for a CQAC query Q and set $\mathcal{V}$ of CQAC views has the minimum number of relational subgoals among all CQAC queries defined using $\mathcal{V}$ that are equivalent (as expansions) to a (unique) CQAC MiCR for Q and $\mathcal{V}$. A globally minimized MiCR may not be unique for a given $(Q, \mathcal{V})$, please see example in the full version [1] of this paper.

While we can show that two distinct minimized MiCRs for a given CQAC MiCR can have a different number of relational (view) subgoals (see [1]), the minimized MiCR output by our algorithm pruned-MiCR is guaranteed to be a *globally* minimized MiCR, see Sect. 4.3 for the details.

4.2 Algorithm for Finding Minimized MiCRs

In this subsection, we present and discuss the pseudocode for our algorithm pruned-MiCR (Algorithm 2). The pseudocode of Algorithm 2 has two parts:

(A) Lines 1 through 11 of the pseudocode present a "full-MiCR" algorithm that outputs a CQAC containing rewriting R of a given CQAC query Q using a given set V of CQAC views. The full version [1] of this paper contains the soundness and completeness result for this specific full-MiCR algorithm when applied to problem inputs such that the homomorphism property (see Sect. 2) holds between the expansion of the MiCR and the input query.
(B) Lines 12 through 28 of the pseudocode present the pruned-MiCR algorithm that is the subject of the discussion in this section of the paper.

Please note that the full-MiCR part (lines 1-11) of Algorithm 2 is not *a contribution of this paper. It is given here just to provide the reader with a complete picture, specifically to indicate which MiCR-generating algorithm was used in our experimental results, see the full version [1] of this paper.*

We now outline the flow of our proposed algorithm pruned-MiCR (lines 12-28 of Algorithm 2). The algorithm accepts as its inputs a CQAC query Q, a set V of CQAC views, and a CQAC MiCR R of Q using V. First (lines 12-22 of the pseudocode) pruned-MiCR constructs buckets, one bucket to represent each (view subgoal, query subgoal) pair, where the views are drawn from the MiCR R, and the query is the input query Q. Suppose two view subgoals g_1 and

Algorithm 1: Algorithm Pruned-MiCR

Input : CQAC query Q, set of CQAC views V
Output: Minimized MiCR of Q using views V
begin

{ Construct the full MiCR (see [1] for a discussion): }
1. $R \leftarrow null$
2. **for** *each view* v *in* V **do**
   3. **for** *each containment mapping* μ_i *from the core subgoals in the body of* v *to* Q **do**
      4. Construct $h(v)$ by replacing each distinguished variable in v with $\mu_i(V)$
      5. $ac \leftarrow null$
      6. $ac\_view \leftarrow AC(h(v))$
      7. **for** *each* $ac_i \in AC(Q)$ **do**
         8. **if** *all variables in* ac_i *appear in* $h(v)$ **then**
            9. $ac \leftarrow ac \wedge ac_i$
      10. **if** $AC(Q) \Rightarrow \mu_i(ac\_view, ac)$ **then**
          11. Add $h(v), ac$ to the rewriting R

{ *pruned-MiCR begins here and ends on line 28:*}
{ Construct buckets:}
12. **for** *each core subgoal* g_r *in* R **do**
    13. **for** *each query subgoal* g_q *that* g_r *maps to* **do**
        14. Let B be the bucket representing g_r, g_q
        15. $ignore\_subgoal \leftarrow false$
        16. **for** *each* g_v *in* B **do**
            17. **if** g_r *strictly contains* g_v **then**
                18. $ignore\_subgoal \leftarrow true$
            19. **else**
                g_v strictly contains g_r
            20. Delete g_v from bucket B
        21. **if** *ignore_subgoal is false* **then**
            22. Add g_r to B

{ Now we have a set of buckets and a set of view subgoals covering each bucket. }
23. Run a minimum set cover algorithm to select a set of view subgoals such that all the buckets are covered.
24. Construct a rewriting by taking a conjunction of the selected views and their associated arithmetic predicates.
25. **if** *candidate rewriting is contained in the full MiCR* **then**
    26. Output candidate rewriting
    27. **else**
        28. Output R { Output the MiCR that was the input of line 12. }.

end

g_2 both cover the same query subgoal and are candidates for the same bucket. Then, in case one of the view subgoals properly contains the other, we keep in the bucket the head homomorphism for the *contained-view* only; otherwise, both view heads are inserted into the bucket. Second (see line 23 of the pseudocode), a minimum set cover algorithm is run to select a subset of the view heads such that each bucket is covered. This set of view heads is used to form a candidate rewriting (line 24 of the pseudocode). Finally (lines 25-28 of the pseudocode), the algorithm checks whether the candidate rewriting is equivalent to the full (input) MiCR R, and outputs the candidate rewriting if the check succeeds. (In case of non-equivalence, pruned-MiCR outputs the full MiCR R.)

Consider an illustration of the flow of the algorithm. We will use the query, views, and CQAC MiCR R of the following example.

Example 3. Let query Q and six views, V_1 through V_6, be defined as follows. Let $Q(X, Z) :- p(X, Y, Y, X, X),\ s(Z, Z),\ Z < 3$; let $V_1() :- p(X_1, A_1, B_1, X_1, C_1)$; let $V_2(X_2) :- p(X_2, A_2, B_2, X_2, C_2)$; let $V_3(X_3) :- p(X_3, A_3, B_3, X_3, X_3)$; let $V_4(X_4) :- p(C_4, A_4, B_4, X_4, X_4)$; let $V_5(B_5) :- p(A_5, Y_5, Y_5, B_5, C_5)$; and let $V_6(Z_6) :- s(Z_6, T_6)$.

It is possible to show that CQAC query $R(X, Z)$:- $V_1(),\ V_2(X),\ V_3(X),\ V_4(X),\ V_5(X),\ V_6(Z),\ Z < 3$. is a CQAC MiCR of Q using $\{V_1, \ldots, V_6\}$. Our algorithm pruned-MiCR generates the following *globally* minimal minimized MiCR: $R'(X, Z)$:- $V_3(X),\ V_5(X),\ V_6(Z),\ Z < 3$. □

In order to minimize the MiCR R to obtain R', algorthm pruned-MiCR retains the views that cover the query subgoals most tightly in the MiCR, and deletes views that cover no query subgoal tightly. Specifically, views V_3, V_5, and V_6 should be retained in the MiCR but not the other views. Views V_3, and V_5 cover the subgoal $p(X, Y, Y, X, X)$ and do not contain each other. At the same time, consider view V_2. V_2 contains view V_3 and thus covers the query less tightly than view V_3. Hence, V_2 should not be present in the minimized MiCR.

4.3 Correctness and Complexity of pruned-MiCR

In this subsection we formulate the correctness and complexity results for our algorithm pruned-MiCR. The proofs and details can be found in [1].

Theorem 6. (Soundness of pruned-MiCR) *Given a CQAC problem input $(Q, \mathcal{V}, R)$, where R is a CQAC MiCR for Q using $\mathcal{V}$, let R' be the (CQAC) output of algorithm pruned-MiCR. Then R' is a globally minimized MiCR for Q and $\mathcal{V}$ whenever R and R' are not isomorphic.*

As suggested by our experimental results, see full version [1] of this paper, for many problem inputs pruned-MiCR outputs minimized MiCRs R' (that are *not* isomorphic to the pruned-MiCR input R, see Theorem 6) whose evaluation costs are significantly lower than those of the input R to the algorithm.

Theorem 7. (Completeness of pruned-MiCR (in the MiCR sense)) *Given a CQAC problem input $(Q, \mathcal{V}, R)$, where R is a CQAC MiCR for Q using*

$\mathcal{V}$, let R' be the (CQAC) output of algorithm pruned-MiCR. Then R' is a CQAC MiCR for Q and $\mathcal{V}$.

While being complete in the sense of Theorem 7, algorithm pruned-MiCR is not complete in the sense that it does not always produce a (globally) *minimized* MiCR for its problem inputs. The reason is that pruned-MiCR does not consider shared variables across query subgoals (i.e., variables that occur in two or more subgoals of the query) while minimizing the MiCR, see [1] for the details.

Finally, the complexity of pruned-MiCR is singly exponential in the size of its problem inputs. See [1] for the details.

References

[1] Afrati, F., Chandrachud, M., Chirkova, R., Mitra, P.: Approximate rewriting of queries using views. Technical Report TR-2009-7, NCSU (2009) Available from `http://www.csc.ncsu.edu/research/tech/reports.php`.

[2] Bayardo, R., Bohrer, W., Brice, R., Cichocki, A., Fowler, J., Helal, A., Kashyap, V., Ksiezyk, T., Martin, G., Nodine, M., Rashid, M., Rusinkiewicz, M., Shea, R., Unnikrishnan, C., Unruh, A., Woelk, D.: InfoSleuth: Semantic integration of information in open and dynamic environments. In: SIGMOD. (1997) 195–206

[3] Halevy, A.: Data integration: A status report. In: BTW. (2003) 24–29

[4] Ullman, J.: Information integration using logical views. Theoretical Computer Science **239**(2) (2000) 189–210

[5] Theodoratos, D., Sellis, T.: Data warehouse configuration. In: VLDB. (1997)

[6] Chaudhuri, S., Krishnamurthy, R., Potamianos, S., Shim, K.: Optimizing queries with materialized views. In: ICDE. (1995) 190–200

[7] Levy, A., Mendelzon, A., Sagiv, Y., Srivastava, D.: Answering queries using views. In: PODS. (1995) 95–104

[8] Abiteboul, S., Duschka, O.: Complexity of answering queries using materialized views. In: PODS. (1998) 254–263

[9] Afrati, F., Chirkova, R., Gergatsoulis, M., Pavlaki, V.: Finding equivalent rewritings in the presence of arithmetic comparisons. In: EDBT. (2006) 941–960

[10] Halevy, A.: Answering queries using views: A survey. VLDB Journal **10(3)** (2001) 270–294

[11] Mitra, P.: An algorithm for answering queries efficiently using views. In: Proceedings of the Australasian Database Conference. (2001)

[12] Pottinger, R., Halevy, A.: MiniCon: A scalable algorithm for answering queries using views. VLDB Journal (2001)

[13] Afrati, F., Li, C., Mitra, P.: Answering queries using views with arithmetic comparisons. In: PODS. (2002)

[14] Tatarinov, I., Halevy, A.: Efficient query reformulation in peer data management systems. In: SIGMOD. (2004) 539–550

[15] Halevy, A., Ives, Z., Madhavan, J., Mork, P., Suciu, D., Tatarinov, I.: The piazza peer data management system. IEEE Transactions on Knowledge and Data Engineering **16**(7) (2004) 787–798

[16] Deutsch, A., Ludäscher, B., Nash, A.: Rewriting queries using views with access patterns under integrity constraints. In: ICDT. (2005) 352–367

[17] Grahne, G., Mendelzon, A.: Tableau techniques for querying information sources through global schemas. In: ICDT. (1999) 332–347

[18] Calì, A., Calvanese, D., Martinenghi, D.: Optimization of query plans in the presence of access limitations. In: EROW. (2007)
[19] Chandra, A., Merlin, P.: Optimal implementation of conjunctive queries in relational data bases. ACM STOC (1977) 77–90
[20] van der Meyden, R.: The complexity of querying indefinite data about linearly ordered domains. In: PODS. (1992) 331–345
[21] Klug, A.: On conjunctive queries containing inequalities. JACM **35**(1) (1988) 146–160
[22] Afrati, F., Li, C., Mitra, P.: Rewriting queries using views in the presence of arithmetic comparisons. Theoretical Computer Science **368**(1-2) (2006) 88–123
[23] Florescu, D., Levy, A., Suciu, D., Yagoub, K.: Optimization of run-time management of data intensive web-sites. In: VLDB. (1999) 627–638
[24] Zaharioudakis, M., Cochrane, R., Lapis, G., Pirahesh, H., Urata, M.: Answering complex SQL queries using automatic summary tables. In: SIGMOD. (2000)
[25] Levy, A., Rajaraman, A., Ordille, J.: Querying heterogeneous information sources using source descriptions. In: VLDB. (1996) 251–262
[26] Afrati, F., Gergatsoulis, M., Kavalieros, T.: Answering queries using materialized views with disjunctions. In: ICDT. (1999) 435–452
[27] Duschka, O., Genesereth, M.: Answering recursive queries using views. In: PODS. (1997) 109–116
[28] Qian, X.: Query folding. In: ICDE. (1996) 48–55
[29] Gupta, A., Sagiv, Y., Ullman, J., Widom, J.: Constraint checking with partial information. In: PODS. (1994) 45–55
[30] Acharya, S., Gibbons, P., Poosala, V., Ramaswamy, S.: The Aqua approximate query answering system. In: SIGMOD. (1999) 574–576
[31] Babcock, B., Chaudhuri, S., Das, G.: Dynamic sample selection for approximate query processing. In: SIGMOD. (2003) 539–550
[32] Chakrabarti, K., Garofalakis, M., Rastogi, R., Shim, K.: Approximate query processing using wavelets. In: VLDB. (2000) 111–122
[33] Poosala, V., Ganti, V., Ioannidis, Y.: Approximate query answering using histograms. IEEE Data Engineering Bulletin **22**(4) (1999) 5–14
[34] Rizvi, S., Mendelzon, A., Sudarshan, S., Roy, P.: Extending query rewriting techniques for fine-grained access control. In: SIGMOD. (2004) 551–562
[35] Miklau, G., Suciu, D.: A formal analysis of information disclosure in data exchange. In: SIGMOD. (2004) 575–586
[36] Miklau, G.: Confidentiality and Integrity in Data Exchange. PhD thesis, University of Washington (2005)
[37] Calvanese, D., Giacomo, G., Lenzerini, M., Vardi, M.: View-based query processing: On the relationship between rewriting, answering and losslessness. In: ICDT. (2005) 321–326
[38] Afrati, F., Li, C., Mitra, P.: On containment of conjunctive queries with arithmetic comparisons. In: EDBT. (2004)
[39] Afrati, F., Li, C., Mitra, P.: On containment of conjunctive queries with arithmetic comparisons (extended version). UCI ICS Technical Report (June 2003)