Query Answering using Views in the Presence of Dependencies

Foto Afrati
Electrical and Computing Engineering
National Technical University of Athens
157 73 Athens, Greece
afrati@cs.ece.ntua.gr

Nikos Kiourtis[*]
Electrical and Computing Engineering
National Technical University of Athens
157 73 Athens, Greece
nkiourt@central.ntua.gr

ABSTRACT

In this paper we investigate equivalent and maximally contained rewritings of a query using views under a set of dependencies. We propose two new algorithms that find equivalent and maximally contained rewritings in the presence of dependencies, and we also prove that a maximally contained rewriting computes all the certain answers under the open world assumption, both in the presence and the absence of dependencies.

1. INTRODUCTION

In many data-management applications, such as information integration, data warehousing, web-site designs, and query optimization, the problem of answering queries using views is of special significance. The presence of dependencies in query answering is also very important, because dependencies are essential for dealing with true concept mismatch between the data of the sources. In this paper, we investigate the problem of finding equivalent and maximally contained rewritings of a query Q using a set of views $\mathcal{V}$ under a set of dependencies $\mathcal{C}$ which is the popular technique for query answering using views.

The problem of finding equivalent rewritings is formally defined as follows: We have a database schema R, a set of CQ views $\mathcal{V}$ over schema R, a set of tgds and egds $\mathcal{C}$ over schema R and a CQ query Q over schema R. The solution to the problem is a conjunctive query Q' over schema of views $\mathcal{V}$ such that $Q \equiv_{\mathcal{C}} Q'$, i.e. for all databases D that satisfy the constraints $\mathcal{C} : Q(D) = Q'(\mathcal{V}(D))$. As it is shown in [4] and [15], such a conjunctive query may not always exist, even in the absence of dependencies. This problem is harder than the problem of finding equivalent rewritings when no dependencies are present, because the presence of some dependencies may assert the equivalence of some rewritings that are otherwise not equivalent if these dependencies are missing. The problem of finding maximally contained rewritings of a query using a set of views is also a recognised open problem (see [3]), and in the presence of dependencies it may be the case that there exist only datalog maximally contained rewritings of a query (see [12] for an example). Our contributions are the following:

- We identify an important property used for optimization in a number of query rewriting algorithms and propose an efficient algorithm that finds equivalent rewritings in the presence of a class of dependencies known as weakly acyclic LAV tgds (section 3).
- We propose an efficient algorithm that finds maximally contained rewritings of a UCQ query Q with respect to UCQ in the presence of weakly acyclic LAV tgds (section 4.1).
- We show that a maximally contained rewriting of a UCQ query Q with respect to UCQ computes the certain answers both in the absence and presence of a set of dependencies $\mathcal{C}$ such that the chase of Q with $\mathcal{C}$ always terminates (section 4.2).

The problem of finding equivalent rewritings was studied in [16] and [8], where an algorithm called Chase & BackChase is presented, and in [11] another algorithm is presented that finds equivalent rewritings under a set of inclusion dependencies. The problem of finding maximally contained rewritings in the presence of functional dependencies or full dependencies was studied in [9], where the inverse rules algorithm was modified to produce rewritings. In [13], an algorithm is presented that finds maximally contained rewritings under a special class of dependencies called conjunctive inclusion dependencies, which are essentially GLAV mappings that are viewed as dependencies. In [7], the authors deal with the problem of creating maximally contained rewritings in GAV data integration systems in the presence of inclusion dependencies and key constraints. The problem of finding contained rewritings in the presence of inclusion dependencies was studied in [6], where the MiniCon algorithm is modified.

2. PRELIMINARIES

A conjunctive query or CQ ([2]) over a schema R is an expression of the form $Q(\vec{x}) : - \phi(\vec{x}, \vec{y})$, where $\phi(\vec{x}, \vec{y})$ is a conjunction of atomic formulas that are also called subgoals of the query. A tuple generating dependency (tgd) is a logic formula of the form $\forall \vec{x}(\phi(\vec{x}) \rightarrow \exists \vec{y} \psi(\vec{x}, \vec{y}))$, and an equality

[*]This research project (PENED 03ED176) is co-financed by E.U.-European Social Fund (75%) and the Greek Ministry of Development-GSRT (25%).

generating dependency (egd) is a logic formula of the form $\forall \vec{x}(\phi(\vec{x}) \rightarrow x_1 = x_2)$. In both tgds and egds, $\phi(\vec{x})$ and $\psi(\vec{x}, \vec{y})$ are conjunctions of atomic formulas over R, all of the variables in $\vec{x}$ must appear in $\phi(\vec{x})$ and x_1, x_2 must appear in $\vec{x}$. Let $\mathcal{C}$ be a set of constraints (tdgs and egds), and Q_1, Q_2 be two conjunctive queries. We say that Q_1 is contained in Q_2 under the constraints $\mathcal{C}$ and we write $Q_1 \sqsubseteq_{\mathcal{C}} Q_2$, if for all databases D that satisfy $\mathcal{C}$ we have that $Q_1(D) \subseteq Q_2(D)$. If $\mathcal{V}$ is a set of views, an equivalent CQ rewriting of Q using $\mathcal{V}$ in the presence of C under the closed world assumption (CWA) is a CQ query R that has the same head variables with Q, its body consists only of views from $\mathcal{V}$ and for all databases D that satisfy $\mathcal{C}$: $R(\mathcal{V}(D)) = Q(D)$. If $\mathcal{L}$ is a query language and $\mathcal{I}$ a view instance, a maximally contained rewriting (MCR) $\mathcal{P}$ of Q with respect to $\mathcal{L}$ using $\mathcal{V}$ in the presence of C under the open world assumption (OWA) is an $\mathcal{L}$–query that uses only views from $\mathcal{V}$ and for all databases D such that $\mathcal{I} \subseteq \mathcal{V}(D)$ and D satisfies the constraints $\mathcal{C}$ we have that $\mathcal{P}(\mathcal{I}) \subseteq Q(D)$ and if there is another $\mathcal{L}$–query $\mathcal{P}'$ such that $\mathcal{P}'(\mathcal{I}) \subseteq Q(D)$, then $\mathcal{P}'(\mathcal{I}) \subseteq \mathcal{P}(\mathcal{I}) \subseteq Q(D)$. We define the certain answers of Q as follows: under CWA, certain$(Q, \mathcal{I}) = \bigcap\{Q(D) : \mathcal{I} = \mathcal{V}(D)\}$, and under OWA, certain$(Q, \mathcal{I}) = \bigcap\{Q(D) : \mathcal{I} \subseteq \mathcal{V}(D)\}$ (in the presence of constraints $\mathcal{C}$, we also require that all databases D used for certain$(Q, \mathcal{I})$ satisfy $\mathcal{C}$, and we write $D \models \mathcal{C}$).

Definition 1. (Chase) Let $\mathcal{C}$ be a set of tgds and egds and let Q be a conjunctive query. A *chase sequence of Q with $\mathcal{C}$* is a finite or infinite sequence of chase steps ([10]) $Q_i \overset{d_i,h_i}{\rightarrow} Q_{i+1}$, $Q_0 = Q$ and d_i a dependency in $\mathcal{C}$. A *finite chase of Q with $\mathcal{C}$* is a finite chase sequence $Q_i \overset{d_i,h_i}{\rightarrow} Q_{i+1}$, $0 \leq i \leq n$, such that there is no dependency $d_j \in \mathcal{C}$ and no homomorphism h_j such that d_j can be applied to Q_n.

In the rest of the paper we assume that UCQ is the language of finite union of conjunctive queries, $\mathcal{V}$ is a set of views in the language of CQs and $\mathcal{I}$ is a view instance for $\mathcal{V}$. Let Q be a CQ, $\mathcal{V}$ a set of views and $\mathcal{C}$ a set of tgds and egds such that the chase sequence of Q with $\mathcal{C}$ always terminates. If $Q \overset{\mathcal{C}}{\rightarrow} Q_{\mathcal{C}}$, then the canonical database D'_Q of $Q_{\mathcal{C}}$ is obtained by turning each subgoal into a fact by replacing each variable in the body by a distinct constant, and treating the resulting subgoals as the only tuples in D'_Q. The set of chased view tuples $\mathcal{T}(Q, \mathcal{V}, \mathcal{C})$ is obtained by taking all the resulting tuples from $\mathcal{V}(D'_Q)$ (i.e. the application of the view definitions $\mathcal{V}$ on D'_Q) and restoring each introduced constant back to the original variable of $Q_{\mathcal{C}}$.

3. CERTAIN ANSWERS IN THE PRESENCE OF DEPENDENCIES UNDER THE CWA

In order to compute the certain answers (both in the presence and absence of dependencies) under the closed world assumption, all we need is an equivalent rewriting of the query. In this section, we deal with the problem of finding equivalent rewritings in the presence of dependencies. The only known algorithm is [8], where the authors find equivalent rewritings in the presence of dependencies with the minimal number of views. It has been proved in the literature that in many cases, the shared variable property (definition 2) speeds up significantly the process of creating such a minimal rewriting, by reducing the search space (for example in Minicon [17], CoreCover [5] and the Shared Variable Bucket algorithm [14]). In this section, we propose a novel alogrithm CoreCover$\mathcal{C}$ that uses the shared variable property to find equivalent rewritings in the presence of dependencies.

We first present a naive algorithm (which is a slight variant of [8]) for finding equivalent rewritings in the presence of a set of weakly–acyclic dependencies $\mathcal{C}$ under the CWA. The naive algorithm works as follows:

1. Chase query Q with $\mathcal{C}$ and create $Q_{\mathcal{C}}$. Compute the canonical database D'_Q of $Q_{\mathcal{C}}$ and the set $\mathcal{T}(Q, \mathcal{V}, \mathcal{C})$
2. Create rewriting R by joining all view tuples in $\mathcal{T}(Q, \mathcal{V}, \mathcal{C})$
3. Compute the expansion R^{exp} of R and chase R^{exp} with $\mathcal{C}$ to create $R^{exp}_{\mathcal{C}}$. Check whether $R^{exp}_{\mathcal{C}} \sqsubseteq Q$. If yes, then R is an equivalent rewriting, or else there is no conjunctive query that is an equivalent rewriting under $\mathcal{C}$

Theorem 1. (Soundness) The rewriting R that is computed from the naive algorithm is an equivalent rewriting of Q under constraints $\mathcal{C}$.

Theorem 2. (Completeness) If there is a conjunctive query R that is an equivalent rewriting of Q under constraints $\mathcal{C}$, then the naive algorithm will find an equivalent rewriting R' of Q under constraints $\mathcal{C}$.

We now define the shared variable property.

Definition 2. **(Shared Variable Property)** Let Q be a conjunctive query with n subgoals, $\mathcal{V}$ be a set of CQ views and $\mathcal{C}$ a set of dependencies such that the chase of Q with $\mathcal{C}$ terminates. Suppose that R is a CQ contained rewriting of Q using $\mathcal{V}$ in the presence of $\mathcal{C}$.

We say that R has the shared variable property if for every homomorphism $h : Q \rightarrow R^{exp}_{\mathcal{C}}$, there is a partition $\{Q_1, \ldots, Q_n\}$ of Q's subgoals and h can be *decomposed* into n homomorphisms $h_1, \ldots, h_n$, in the following sense:

- $h_i : Q_i \rightarrow (g^{exp}_i)'$, where g_i is the i-th subgoal of R and $(g^{exp}_i)'$ is the set of atoms we obtain after chasing the expansion of g_i with $\mathcal{C}$.
- For every variable x defined in h_i, $h_i(x) = h(x)$.

Unfortunately, as the next theorem shows, there are cases when $\mathcal{C} \neq \emptyset$ and the shared variable property does not hold for any of the candidate rewritings.

Theorem 3. There exists a query Q, a view set $\mathcal{V}$ and a set of functional dependencies $\mathcal{C}$ such that for every equivalent rewriting of Q using $\mathcal{V}$ under $\mathcal{C}$ the shared variable property does not hold.

We propose a novel algorithm CoreCover$\mathcal{C}$ for finding equivalent rewritings in the presence of a set of tgds such that all equivalent rewritings have the shared variable property. First we revisit the definition of the tuple–core from [5]:

Definition 3. Let $t_v \in \mathcal{T}(Q, \mathcal{V}, \mathcal{C})$. A chased tuple–core of t_v is a maximal collection G of subgoals in the query Q such that there is a containment mapping μ from G to $(t^{exp}_v)'$ of t_v, where $(t^{exp}_v)'$ is the chased expansion of t_v with $\mathcal{C}$, and μ has the following properties:

1. μ is a one–to–one mapping, and it maps the arguments in G that appear in t_v as the identity mapping on arguments

2. Each distinguished variable X in G_v is mapped to a distinguished variable in t_v^{exp} (furthermore, by property (1), $\mu_{t_v}(X) = X$)

3. If a nondistinguished variable X in G is mapped under μ to an existential variable in t_v's expansion, then G includes all subgoals in Q that use this variable X.

We now describe CoreCover$\mathcal{C}$:

1. Chase query Q with $\mathcal{C}$ and create $Q_{\mathcal{C}}$. Compute the canonical database D'_Q of $Q_{\mathcal{C}}$ and the set $\mathcal{T}(Q, \mathcal{V}, \mathcal{C})$. Compute the chased tuple cores for every $t_v \in \mathcal{T}(Q, \mathcal{V}, \mathcal{C})$

2. Find a minimum number of view tuples to cover the query's subgoals and create R by joining them

This algorithm works like the naive one, but it uses the chased tuple–cores to find a minimum number of view tuples that cover the query subgoals (this can be done using an algorithm for the set–covering problem). Notice that by the construction of the tuple–cores, we do not need to chase or to perform an additional containment mapping check as before. We next prove that CoreCover$\mathcal{C}$ is sound and complete.

Theorem 4. Algorithm CoreCover$\mathcal{C}$ is sound and complete for the class of tgds such that all equivalent rewritings have the shared variable property.

Definition 4. We define $\mathcal{C}^w_{\mathsf{LAV}}$ to be the set of LAV (Local–As–View) tgds that consists only of weakly–acyclic tgds ([10]) of the form $\forall\vec{x}(A(\vec{x}) \rightarrow \exists\vec{y}\psi(\vec{x}, \vec{y}))$, i.e. where the left hand side consists of a single atom $A(\vec{x})$.

It is clear that $\mathcal{C}^w_{\mathsf{LAV}}$ contains all weakly–acyclic inclusion dependencies. We next prove that the shared variable property will hold on all equivalent rewritings under a set $\mathcal{C} \subseteq \mathcal{C}^w_{\mathsf{LAV}}$:

Theorem 5. Let Q be a CQ and $\mathcal{V}$ a set of CQ views. If R is a contained rewriting of Q using $\mathcal{V}$ under a set of tgds $\mathcal{C} \subseteq \mathcal{C}^w_{\mathsf{LAV}}$, then R has the shared variable property.

Corollary 1. CoreCover$\mathcal{C}$ is sound and complete for $\mathcal{C}^w_{\mathsf{LAV}}$.

4. CERTAIN ANSWERS IN THE PRESENCE OF DEPENDENCIES UNDER OWA

4.1 Algorithm for finding MCRs

In this section we give an efficient algorithm which finds a maximally contained rewriting with respect to UCQ of a UCQ Q in the presence of a set of tgds $\mathcal{C} \subseteq \mathcal{C}^w_{\mathsf{LAV}}$ if such an MCR exists, since in the presence of dependencies an MCR of Q with respect to UCQ may not always exist (see [12] for an example). The inverse rules algorithm [9] also finds MCRs but in the the presence of full and functional dependencies, and it creates Datalog MCRs, whereas our algorithm creates UCQ MCRs (if they exist).

We first give some definitions from [3]. A *subgoal mapping* is a mapping from the query subgoals to view subgoals of a view such that the predicate names match. A subgoal mapping induces an *associated argument mapping*, which maps each query variable/constant to a variable/constant in the body of the view definition, such that for each query subgoal g that is mapped to a view subgoal, their variables and constants are also mapped argument-wise. Notice that an argument mapping is not restricted to map a query variable/constant to a single view variable/constant (as in a containment mapping), since it may map a query variable/constant to several view variables/constants. Given an argument mapping, we associate with it several containment mappings. An *associated containment mapping* is a mapping from query variables/constants to view variables/ constants defined by a partition P on the set of the view variables/constants into equivalence classes, in such a way that each query variable/constant is mapped to elements of a single equivalence class, all variables/constants of a query subgoal are mapped on the variables/constants of a single copy of a view, and the following three conditions hold: (a) each equivalence class with more than one element is populated by either (identical) constants and/or distinguished variables; (b) an equivalence class which is the image of a constant has only distinguished variables (even if it contains only one element) and possibly the same constant. (c) Distinguished variables map to distinguished variables.

Given a total subgoal mapping and one of its associated containment mappings (if there exists any), we can define a conjunctive query over view subgoals that uses the view copies that are involved in the associated containment mapping, where the distinguished view variables are equated according to the partition that defines the associated containment mapping. We now present MINICON$\mathcal{C}$:

1. Chase each view $V \in \mathcal{V}$ with $\mathcal{C}$ and create $V_{\mathcal{C}}$. Let $\mathcal{V}_{\mathcal{C}}$ be the set of the chased views. Set $\mathcal{P} = \emptyset$

2. Generate the set of MCDs ([17]) $\mathcal{M}$ as follows:

   For each query subgoal g in Q, for each view $V \in \mathcal{V}_{\mathcal{C}}$ and for each subgoal g' in V, try to find a containment mapping $\mu : g \rightarrow g'$ and if it exists, add to $\mathcal{M}$ a new partial MCD: $(\{g\}, \mu)$. While there are partial MCDs with shared variables, for each partial MCD (G, μ) do the following: Choose a shared variable X in G and a query subgoal g not in G that contains X. For each view subgoal g' that has an argument mapping μ' from g, extend μ with μ' and create μ'', then replace the current partial MCD with new partial MCD: $(G' = G \cup \{g\}, \mu'')$

3. Combine the set of MCDs $\mathcal{M}$ as follows:

   For each combination of MCDs that covers all query subgoals without overlapping (i.e. the combination of MCD mappings is a total subgoal mapping), let μ be the corresponding argument mapping. Check whether there exists an associated containment mapping. If it does exist, then find the most relaxed associated containment mapping as follows: For each query variable/constant X form a class that contains all view variables/constants that are images of X under μ, and while classes are not disjoint merge classes that share an element. If there is a class containing two distinct constants this procedure fails, else return the "relaxed" classes as the most relaxed associated containment mapping μ'. Use μ' to create a contained rewriting R. Set $\mathcal{P} := \mathcal{P} \cup R$

4. Return the UCQ MCR $\mathcal{P}$

In the absence of constraints, MINICON$\mathcal{C}$ reduces to the Minicon algorithm [17]. We next prove that the algorithm will always find an MCR in the language of finite unions of conjunctive queries, if such an MCR exists.

Theorem 6. Let Q be a UCQ, $\mathcal{V}$ be a set of views in the language of CQs and $\mathcal{C} \subseteq \mathcal{C}^{w}_{\mathsf{LAV}}$ be a set of tgds. Suppose there exists a maximally contained rewriting of Q with respect to UCQ using $\mathcal{V}$ in the presence of $\mathcal{C}$. Then, MiniconC outputs a maximally contained rewriting of Q with respect to UCQ using $\mathcal{V}$ in the presence of $\mathcal{C}$.

4.2 MCRs vs Certain Answers

The first observation about the connection of MCRs and certain answers appeared in [1], where Datalog is assumed as the language of rewriting. Here we first extend this result (theorem 7) to capture any rewriting language. Then, in theorem 8 we prove a weaker result like that of theorem 7 in the presence of dependencies.

Theorem 7. Let Q be a UCQ query, $\mathcal{V}$ a set of views and $\mathcal{P}$ an MCR of Q with respect to UCQ. Let $\mathcal{I}$ be a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$. Then, under the open world assumption, $\mathcal{P}$ computes all the certain answers of Q in any view instance $\mathcal{I}$: $\mathcal{P}(\mathcal{I}) = \text{certain}(Q, \mathcal{I})$

Corollary 2. Let Q be a UCQ query, $\mathcal{V}$ be a set of views and $\mathcal{P}$ be a MCR of Q wrt UCQ. If $\mathcal{I}$ is a view instance such that $certain(Q, \mathcal{I}) \neq \emptyset$, then $\mathcal{P}(\mathcal{I}) = \text{certain}(Q, \mathcal{I})$.

The proof of theorem 7 uses the following lemma:

Lemma 1. Let Q be CQ query and $\mathcal{V}$ be a set of CQ views. Let $\mathcal{I}$ be a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$. Then there is a finite space $\mathcal{S}$ of contained rewritings (of Q using $\mathcal{V}$) of size which is a function only of the sizes of the views and the query, such that the following happens: Given a tuple $t_0 \in certain(Q, \mathcal{I})$, there is a contained CQ rewriting R in $\mathcal{S}$ such that $t_0 \in R(\mathcal{I})$.

In the presence of dependencies $\mathcal{C}$, we need to state and prove a weaker lemma than lemma 1 to reflect the changes introduced by $\mathcal{C}$. This is necessary due to the fact that in this case, a UCQ MCR may not exist in general, as we already argued in the beginning of the section.

Lemma 2. Let Q be CQ query, $\mathcal{V}$ be a set of CQ views, $\mathcal{C}$ be a set of dependencies such that the chase of Q with $\mathcal{C}$ terminates and $\mathcal{P}$ be an UCQ MCR of Q wrt UCQ. Let $\mathcal{I}$ be a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$ and $D \models \mathcal{C}$. Then there is a finite space $\mathcal{S}'$ of contained rewritings (of Q using $\mathcal{V}$) of size which is a function of the sizes of $\mathcal{P}$ and $\mathcal{V}$, such that the following happens: Given a tuple $t_0 \in certain(Q, \mathcal{I})$, there is a contained CQ rewriting R in $\mathcal{S}'$ such that $t_0 \in R(\mathcal{I})$.

Theorem 8. Let Q be a UCQ query, $\mathcal{V}$ be a set of views, $\mathcal{C}$ be a set of weakly acyclic tgds and egds and $\mathcal{P}$ be a MCR of Q under $\mathcal{C}$ with respect to UCQ. Let $\mathcal{I}$ be a view instance such that there exists a database instance D such that $\mathcal{I} \subseteq \mathcal{V}(D)$ and $D \models \mathcal{C}$. Then, under the Open World Assumption, $\mathcal{P}$ computes all the certain answers of Q in any view instance $\mathcal{I}$, i.e. $\text{certain}(Q, \mathcal{I}) = \mathcal{P}(\mathcal{I})$.

5. REFERENCES

[1] S. Abiteboul and O. M. Duschka. Complexity of answering queries using materialized views. In *PODS '98: Proceedings of the seventeenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 254–263, New York, NY, USA, 1998. ACM.

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

[3] F. Afrati, C. Li, and P. Mitra. Answering queries using views with arithmetic comparisons. In *PODS '02: Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 209–220, New York, NY, USA, 2002. ACM.

[4] F. N. Afrati. Rewriting conjunctive queries determined by views. In *MFCS*, pages 78–89, 2007.

[5] F. N. Afrati, C. Li, and J. D. Ullman. Generating efficient plans for queries using views. In *SIGMOD '01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data*, pages 319–330, New York, NY, USA, 2001. ACM.

[6] Q. Bai, J. Hong, and M. F. McTear. Query rewriting using views in the presence of inclusion dependencies. In *WIDM '03: Proceedings of the 5th ACM international workshop on Web information and data management*, pages 134–138, New York, NY, USA, 2003. ACM.

[7] A. Calì, D. Lembo, and R. Rosati. Query rewriting and answering under constraints in data integration systems. In *IJCAI-03*, pages 16–21. MK, 2003.

[8] A. Deutsch, L. Popa, and V. Tannen. Query reformulation with constraints. *SIGMOD Records*, 35(1):65–73, 2006.

[9] O. M. Duschka, M. R. Genesereth, and A. Y. Levy. Recursive query plans for data integration. *Journal of Logic Programming*, 43(1):49–73, 2000.

[10] R. Fagin, P. G. Kolaitis, R. J. Miller, and L. Popa. Data exchange: semantics and query answering. *Theoretical Computer Science*, 336(1):89–124, May 2005.

[11] J. Gryz. Query rewriting using views in the presence of functional and inclusion dependencies. *Inf. Syst.*, 24(7):597–612, 1999.

[12] A. Y. Halevy. Answering queries using views: A survey. *The VLDB Journal*, 10(4):270–294, 2001.

[13] C. Koch. Query rewriting with symmetric constraints. In *FoIKS '02: Proceedings of the Second International Symposium on Foundations of Information and Knowledge Systems*, pages 130–147, London, UK, 2002. Springer-Verlag.

[14] P. Mitra. An algorithm for answering queries efficiently using views. In *ADC '01: Proceedings of the 12th Australasian database conference*, pages 99–106, Washington, DC, USA, 2001. IEEE Computer Society.

[15] A. Nash, L. Segoufin, and V. Vianu. Determinacy and rewriting of conjunctive queries using views: A progress report. In *ICDT*, pages 59–73, 2007.

[16] L. Popa, A. Deutsch, A. Sahuguet, and V. Tannen. A chase too far? In *Proceedings of the ACM SIGMOD Conference*, pages 273–284, 2000.

[17] R. Pottinger and A. Halevy. Minicon: A scalable algorithm for answering queries using views. *The VLDB Journal*, 10(2-3):182–198, 2001.