Complexity of Answering Queries Using Materialized Views

(Extended Abstract)

Serge Abiteboul*
INRIA-Rocquencourt
78153 Le Chesnay Cedex, France
phone: (33-1) 39 63 55 37
fax: (33-1) 39 63 53 30
Serge.Abiteboul@inria.fr

Oliver M. Duschka
Stanford University
Gates Building #4B, Rm. 490
Stanford, CA 94305
phone: (650) 725-3110
fax: (650) 725-4671
duschka@cs.stanford.edu

Abstract

We study the complexity of the problem of answering queries using materialized views. This problem has attracted a lot of attention recently because of its relevance in data integration. Previous work considered only conjunctive view definitions. We examine the consequences of allowing more expressive view definition languages. The languages we consider for view definitions and user queries are: conjunctive queries with inequality, positive queries, datalog, and first-order logic. We show that the complexity of the problem depends on whether views are assumed to store all the tuples that satisfy the view definition, or only a subset of it. Finally, we apply the results to the view consistency and view self-maintainability problem which arise in data warehousing.

*Part of the work performed when the author was visiting Stanford University.

1 Introduction

The notion of materialized view is essential in databases [Ull88] and is attracting more and more attention with the popularity of data warehouses [LZW+97]. The problem of answering queries using materialized views [LMSS95, CKPS95, DG97a, BLR97, YL87, RSU95, LRU96, Ull97, DL97, Dus97, DG97b, LRO96] has been studied intensively. We propose a systematic study of its complexity. We also briefly consider the related problems of view consistency and view self-maintainability [Huy97]. Our results exhibit strong connections with two among the most studied problems in database theory, namely query containment [CM77, SY80, Klu88, Shm87, CV92, JK84, DS96, LS97] and incomplete information querying, e.g. [IJ84, AKG91]. Indeed, the works most closely related to our complexity results are perhaps that of van der Meyden [vdM92, vdM93, vdM97] and Vardi [Var82] on (indefinite) database queries. Our results highlight the basic roles played by negation (and in its weak form inequality) and recursion, and a crucial difference between *open* and *closed world assumption* in the view definition.

The main focus of the paper is the study of the *data complexity* of the problem of *answering queries using materialized views*. More precisely, the problem is for a fixed view definition and a fixed query, given a view instance I and a tuple t, is t a *certain* answer, i.e. is t in the answer to the query on the database no matter which is the database yielding the view instance I. This articulation of the problem highlights the main parameters: (i) What are the database and the view models? (ii) What are the query and the view definition languages? (iii) Is *yielding* assuming an open or a closed world?

In the present paper, we use the relational model for the database and the view model. However, our work strongly suggests moving towards an incomplete information model, e.g. conditional tables [IJ84]. Indeed, we will briefly show how these tables can be used for solving the problem (in most solvable cases). For the query and view definition languages, we consider the most popular formal query languages, namely, conjunctive queries (with inequality), positive queries, datalog, and first order logic. We focus on *certain* answers, i.e. tuples that are in the answer for *any* database yielding this particular view instance.

Not surprisingly, our results indicate that recursion and negation in the view definition lead to undecidability. Somewhat also expectedly, we show that the closed world assumption sharply complicates the problem. For instance, under the open world assumption the certain answers in the conjunctive view definitions/datalog queries case can be computed in polynomial time. On the other hand, already the conjunctive view definitions/conjunctive queries case is co-NP-complete under the closed world assumption. This is an a-posteriori argument for a number of recent works that postulate an open world interpretation of views. Perhaps more unexpectedly, we prove that inequalities (a very weak form of negation) lead to intractability. Even under the open world assumption, adding inequalities to the queries, or disjunction to the view definitions make the problem co-NP-hard.

2 The problem

In this section, we present the problem. We assume some familiarity with database theory [Ull88, AHV95]. We start with a database instance D, a view definition $\mathcal{V}$, and a view instance I. The database consists of a set of relations and so does the view. Now, given a query $\mathcal{Q}$, we would like to compute $\mathcal{Q}(D)$. However, we assume that we ignore D and only have access to I, so we will try to get the best possible estimate of $\mathcal{Q}(D)$ given I.

Let us be more precise. Under the *closed world assumption* (CWA), the view instance I stores *all* the tuples that satisfy the view definitions in $\mathcal{V}$, i.e. $I = \mathcal{V}(D)$. Under the *open world assumption* (OWA), on the other hand, instance I is possibly incomplete and might only store *some* of the tuples that satisfy the view definitions in $\mathcal{V}$, i.e. $I \subseteq \mathcal{V}(D)$. As we can see from the following example, in reasoning about the underlying database, it makes a difference whether we are working under the open or closed world assumption.

Example 2.1 Consider the following view definition where the view consists of two relations:

$$v_1(X) :- p(X,Y)$$
$$v_2(Y) :- p(X,Y)$$

and assume that the view instance consists of $\{v_1(a),\ v_2(b)\}$. Under OWA, we only know that some p tuple has value a as its first component, and some (possibly different) p tuple has value b as its second component.

Under CWA, however, we can conclude that all p tuples have value a as their first component and value b as their second component, i.e. p contains exactly the tuple $\langle a, b\rangle$. □

Given some view definition and a view instance, observe that there may be a number of possible databases, i.e. database instances that yield this view instance for this view definition. So, we can think of the database as the *incomplete database* [IJ84] consisting of this set of possible databases. To answer a query, we focus on *certain* answers, i.e. on tuples that are in the answer for each possible database. As seen in Example 2.1, this depends on whether we are assuming an open or a closed world. Indeed, an answer that is certain under OWA is also certain under CWA, but the converse does not hold in general. For instance, in the previous example, the query "is $\langle a, b\rangle$ certainly in p?" is answered positively under CWA and negatively under OWA. In fact, we will show that computing certain answers under CWA is harder than under OWA. The following definition formalizes the concept of certain answer under both assumptions:

Definition 2.1 (certain answer) Let $\mathcal{V}$ be a view definition, I be an instance of the view, and $\mathcal{Q}$ a query. A tuple t is a *certain answer under OWA* if t is an element of $\mathcal{Q}(D)$ for each database D with $I \subseteq \mathcal{V}(D)$. A tuple t is a *certain answer under CWA* if t is an element of $\mathcal{Q}(D)$ for each database D with $I = \mathcal{V}(D)$. □

To conclude this section, we briefly recall the query languages we consider and the standard notion of complexity we use.

Query and view languages A *datalog rule* is an expression of the form:

$$p(\bar{X}) :- p_1(\bar{X}_1), \ldots, p_n(\bar{X}_n)$$

where p, and $p_1, \ldots, p_n$ are predicate names, and $\bar{X}$, $\bar{X}_1$, …, $\bar{X}_n$ are tuples of variables and constants. Each variable in the head of a rule must also occur in the body of the rule. A *datalog* query is a finite set of datalog rules. The notion of *recursive* datalog query/rule is defined in the standard way. A *conjunctive query* (CQ) is a single non-recursive datalog rule. If the body of a conjunctive query is allowed to contain the inequality predicate ($\neq$), then the query is called a *conjunctive query with inequality* ($CQ^{\neq}$). Every variable in a query with inequality must occur at least once in a relational predicate. A *positive query* (PQ) is a non-recursive datalog query together with one particular predicate defined by the query. The query language $PQ^{\neq}$ is obtained by also allowing $\neq$. Finally, *first-order queries* (FO) are defined in the standard way.

A *materialized view*, also called *view instance*, is the stored result of previously executed queries. A *view definition* $\mathcal{V}$ therefore consists of a set of queries defining a finite set of predicates. So, for a query language $\mathcal{L}$, we write $\mathcal{V} \subseteq \mathcal{L}$ to denote the fact that each predicate in the view is defined using a query in $\mathcal{L}$.

Data complexity We will be interested in the *data complexity* of the problem of computing certain answers under the open and closed world assumption. The *data complexity* is the complexity of the problem as a function of the size of the view instance. We will also refer to the *query* and *combined complexity* of the problem. The *query complexity* is the complexity of the problem as a function of the size of the view definition $\mathcal{V}$ and the query $\mathcal{Q}$. The *combined complexity* is the complexity of the problem as a function of these two arguments plus the size of the view instance. (These three notions are due to [Var82].) In the remaining of the paper, when we discuss complexity, we always mean data complexity unless specified otherwise.

In Section 3, we prove that the problem is in co-NP for a wide range of cases. We also highlight some connections with conditional table querying. In Section 4, we examine the complexity of the problem of computing certain answers under OWA and in Section 5 under CWA. In Section 6, we consider view self-maintainability and consistency. Due to space limitations, proofs are omitted or simply sketched. Full proofs can be found in [AD97].

3 Using Conditional Tables

In this section, we briefly sketch a solution to the problem for the open and closed world assumption, when the view is defined in $PQ^{\neq}$ and the query is in $datalog^{\neq}$. We also present an effective procedure that is based on conditional tables [IJ84] that were introduced to represent incomplete information. Indeed, a main purpose of this section is to highlight the strong links between our problem and that of querying incomplete databases.

First we see next that, for $PQ^{\neq}$ views and $datalog^{\neq}$ queries, the problem is in co-NP. So within these limits, it will suffice in the following of the paper to prove co-NP-hardness to establish co-NP-completeness.

Theorem 3.1 *For $\mathcal{V} \subseteq PQ^{\neq}$, $\mathcal{Q} \in datalog^{\neq}$, the problem of determining, given a view instance, whether a tuple is a certain answer under OWA or CWA, is in co-*NP.

Proof. (sketch) To prove that t is not in the answer, it suffices to guess a polynomial-size database D, check whether it produces view instance I, and check that $t \notin \mathcal{Q}(D)$. □

We next turn to an effective way of computing the certain answers. The intuition is to represent, given a view instance, the set of possible databases (or more precisely a sufficient set of possible databases) by a conditional table and then query the conditional table using the techniques introduced by Imielinski and Lipski [IJ84]. Due to space limitations, we refer to [IJ84, AHV95] for a definition of conditional tables. Intuitively, a conditional table is a database instance which might have variables as entries in its tuples. There is also a global condition [AHV95] on the set of variables and for each tuple, a local condition controlling the actual presence of the tuple. A *possible database* for a table T is obtained by choosing a valuation satisfying the global condition, keeping only those tuples with a true local condition and valuating the variables in those tuples.

The following result shows how the problem of querying materialized views can be reduced to that of querying conditional tables, thereby highlighting the strong connection between materialized views and incomplete databases. (The result and the construction used in its proof are then illustrated by examples.)

Theorem 3.2 *Let $\mathcal{V} \subseteq PQ^{\neq}$ and I a view instance. Then one can construct a conditional table (with global condition) T_{owa}, resp. T_{cwa}, such that for each $datalog^{\neq}$ query $\mathcal{Q}$, the certain answers to $\mathcal{Q}$ using view instance I under OWA, resp. CWA, are exactly the certain answers to $\mathcal{Q}$ given the incomplete database specified by T_{owa}, resp. T_{cwa}.*

The previous theorem provides a (co-NP) means of evaluating a query on a database given some materialized view: compute the conditional table and then evaluate the query on the table using [IJ84] techniques.

Example 3.1 Suppose the view is specified by:

$$\begin{array}{l} v(0,Y) \ :- \ p(0,Y) \\ v(X,Y) :- \ p(X,Z),\ p(Z,Y) \end{array}$$

and the view instance consists of $\{v(0,1), v(1,1)\}$. Then there are 2 different ways to obtain the first tuple and only one for the second. This yields the following conditional table for p (the global condition is *true*):

0	1	$w = 1$
0	z	$w \neq 1$
z	1	$w \neq 1$
1	u	*true*
u	1	*true*

This is the table needed for OWA. For CWA, we have to introduce the constraints that the view does not hold for any other tuple. One finds the following (complete) table for p:

0	1	*true*
1	1	*true*

□

Due to space limitations, we do not give a proof of the previous result. However, we briefly discuss some aspects of the construction of the conditional table. Consider the simplest case, i.e. a conjunctive query view under OWA. Intuitively, the table is constructed by "skolemizing" the variables in the conjunctive query in a

(a) open world assumption

views	— query — CQ	$CQ^{\neq}$	PQ	$datalog$	FO
CQ	PTIME	CO-NP	PTIME	PTIME	undec.
$CQ^{\neq}$	PTIME	CO-NP	PTIME	PTIME	undec.
PQ	CO-NP	CO-NP	CO-NP	CO-NP	undec.
$datalog$	CO-NP	undec.	CO-NP	undec.	undec.
FO	undec.	undec.	undec.	undec.	undec.

(b) closed world assumption

views	— query — CQ	$CQ^{\neq}$	PQ	$datalog$	FO
CQ	CO-NP	CO-NP	CO-NP	CO-NP	undec.
$CQ^{\neq}$	CO-NP	CO-NP	CO-NP	CO-NP	undec.
PQ	CO-NP	CO-NP	CO-NP	CO-NP	undec.
$datalog$	undec.	undec.	undec.	undec.	undec.
FO	undec.	undec.	undec.	undec.	undec.

Figure 1: Complexity of computing certain answers

standard manner such as [End72]. Now, we obtain a conditional table. This is not quite a representation of the possible databases since a possible database may contain additional tuples. But with respect to *certain* answers, we can simply query this conditional table in the style of [IJ84] and get the desired certain answers.

Now consider disjunctions in the view definition, we use tuple local conditions as done in the example (with $w = 1$ and $w \neq 1$). Finally for CWA, this is done by evaluating the view definition $\mathcal{V}$ on the conditional table corresponding to OWA and adding as a constraint that each tuple it generates is indeed in the view instance I. In the example, the conditions simplify dramatically, but in general, this may result in rather gory tables. Note that, more generally, one could similarly introduce any *total dependency* [FV86, Var87] on the database by *chasing* [ASU79b, ASU79a] the conditional table as in [IJ84, Gra91]. Observe also that from a practical viewpoint, this raises the issue of obtaining practical restrictions that prevent the conditions from becoming too complicated.

4 Open world assumption

Figure 1 (a) gives an overview of the complexity of computing certain answers under OWA. Under OWA, the problem of computing certain answers is closely related to the query containment problem. Therefore, decidability and undecidability results carry over in both directions. As shown in Theorem 4.1, if the problems are decidable, then their *query complexity* is the same.

Theorem 4.1 *Let $\mathcal{L}_1, \mathcal{L}_2 \in \{CQ, CQ^{\neq}, PQ, datalog, FO\}$ be a view definition language and query language respectively. Then the problem of computing certain answers under OWA of a query $\mathcal{Q} \in \mathcal{L}_2$ given a view definition $\mathcal{V} \subseteq \mathcal{L}_1$ and a view instance is decidable if and only if the containment problem of a query in $\mathcal{L}_1$ in a query in $\mathcal{L}_2$ is decidable. Moreover, if the problems are decidable then the combined complexity of the view problem and the query complexity of the containment problem are identical, so the data complexity of the problem of computing certain answers under OWA is at most the query complexity of the query containment problem.*

Proof. (sketch) The claim is established by giving reductions between the two problems in both directions. We consider here only the reduction from the problem of computing certain answers under OWA to the query containment problem. Let $\mathcal{V} = \{v_1, \ldots, v_k\} \subseteq \mathcal{L}_1$ be a view definition, $\mathcal{Q} \in \mathcal{L}_2$ a query, I a view instance, and t a tuple of the same arity as the head of $\mathcal{Q}$. Let $\mathcal{Q}'$ be a query consisting of the rules of definition $\mathcal{V}$ together with the rule

$$q'(t) :- v_1(t_{11}),\ \ldots,\ v_1(t_{1n_1}), \ldots, v_k(t_{k1}),\ \ldots,\ v_k(t_{kn_k})$$

where I is the instance $v_1 = \{t_{11}, \ldots, t_{1n_1}\}, \ldots, v_k = \{t_{k1}, \ldots, t_{kn_k}\}$. If $\mathcal{L}_1$ is CQ or $CQ^{\neq}$, then the view definitions in $\mathcal{V}$ can be substituted in for the view literals in this new rule. This yields just one conjunctive query. In all cases, $\mathcal{Q}'$ is in $\mathcal{L}_1$. One can show that tuple t is a certain answer of $\mathcal{Q}$ given $\mathcal{V}$ and I if and only if $\mathcal{Q}'$ is contained in $\mathcal{Q}$. The converse is similar. □

The previous theorem involves query complexity. However, we are primarily concerned by data complexity and query complexity results can be misleading. For example, the query complexity of the containment problem of a conjunctive query in a datalog query is EXPTIME-complete, whereas the containment problem of a conjunctive query in a conjunctive query with inequality is considerably easier, namely Π_2^p-complete [vdM97]. In comparison, the data complexity of the problem of computing certain answers under OWA for conjunctive view definitions and datalog queries is polynomial, whereas it is considerably harder, namely co-NP-complete, for conjunctive view definitions and conjunctive queries with inequality.

4.1 Conjunctive view definitions

In this section we consider the complexity of the problem of computing certain answers under OWA in the case of conjunctive view definitions. We will consider queries of different expressive power.

Polynomial cases The main tool for proving polynomial time bounds is the notion of maximally-contained query plans. We recall the relevant definitions here.

The input of a datalog query $\mathcal{Q}$ consists of a database D storing instances of all EDB predicates in $\mathcal{Q}$. Given such a database D, the output of $\mathcal{Q}$, denoted $\mathcal{Q}(D)$, is an instance of the query predicate $\mathcal{Q}$ as determined by, for example, naive evaluation [Ull89]. A datalog query $\mathcal{Q}'$ is *contained* in a datalog query $\mathcal{Q}$ if, for all databases D, $\mathcal{Q}'(D)$ is contained in $\mathcal{Q}(D)$.

A datalog query $\mathcal{P}$ is a *query plan* if all EDB predicates in $\mathcal{P}$ are view literals. The *expansion* $\mathcal{P}^{exp}$ of a datalog query plan $\mathcal{P}$ is obtained from $\mathcal{P}$ by replacing all view literals with their definitions. Existentially quantified variables in view definitions are replaced by new variables in the expansion. A query plan $\mathcal{P}$ is *maximally-contained* in a datalog query $\mathcal{Q}$ w.r.t. a view definition $\mathcal{V}$ if $\mathcal{P}^{exp} \subseteq \mathcal{Q}$, and for each query plan $\mathcal{P}'$ with $(\mathcal{P}')^{exp} \subseteq \mathcal{Q}$, it is the case that $\mathcal{P}'$ is also contained in $\mathcal{P}$. Intuitively, a maximally-contained query plan is the best of all datalog query plans in using the information available from the view instances. We showed in [DG97a] that it is easy to construct these maximally-contained query plans in the case of conjunctive view definitions.

Theorem 4.2 shows that maximally-contained query plans compute exactly the certain answers under OWA.

Theorem 4.2 *For $\mathcal{V} \subseteq CQ$, $\mathcal{Q} \in datalog$, and query plan $\mathcal{P}$ that is maximally-contained in $\mathcal{Q}$ with respect to $\mathcal{V}$, $\mathcal{P}$ computes exactly the certain answers of $\mathcal{Q}$ under OWA for each view instance I.*

As shown in [DG97a] for all $\mathcal{V} \subseteq CQ$ and $\mathcal{Q} \in datalog$, corresponding maximally-contained datalog query plans can be constructed. Because the data complexity of evaluating datalog queries is polynomial [Var82], this implies that the problem of computing certain answers under OWA can be done in polynomial time.

Corollary 4.1 *For $\mathcal{V} \subseteq CQ$ and $\mathcal{Q} \in datalog$, the problem of computing certain answers under OWA can be done in polynomial time.*

Inequalities in the view definition We next show (Theorem 4.3) that adding inequalities just to the view definition doesn't add any expressive power. The certain answers are exactly the same as if the inequalities in the view definition were omitted. This means that the maximally-contained datalog query constructed from the query and the view definition but disregarding the inequality constraints computes exactly the certain answers. Therefore, the problem remains polynomial. (The proof is by case analysis.)

Theorem 4.3 *Let $\mathcal{V} \subseteq CQ^{\neq}$ and $\mathcal{Q} \in datalog$. Define $\mathcal{V}^-$ to be the same view definitions as in $\mathcal{V}$ but with the inequality constraints deleted. Then a tuple t is a certain answer under the open world assumption given $\mathcal{V}$, $\mathcal{Q}$ and a view instance I if and only if t is a certain answer under OWA given $\mathcal{V}^-$, $\mathcal{Q}$ and I.*

Inequalities in the query On the other hand, we see next (Theorem 4.4) that adding inequalities to queries does add expressive power. A single inequality in a conjunctive query, even combined with purely conjunctive view definitions, suffices to make the problem co-NP-hard. Van der Meyden proved a similar result [vdM92], namely co-NP hardness for the case $\mathcal{V} \subseteq CQ^{<}$ and $\mathcal{Q} \in CQ^{<}$. Our theorem strengthens this result to $\mathcal{V} \subseteq CQ$ and $\mathcal{Q} \in CQ^{\neq}$.

Theorem 4.4 *For $\mathcal{V} \subseteq CQ$, $\mathcal{Q} \in CQ^{\neq}$, the problem of determining whether, given a view instance, a tuple is a certain answer under OWA is co-NP-hard.*

Proof. (sketch) Let φ be a 3CNF formula with variables $x_1, \ldots, x_n$ and conjuncts $c_1, \ldots, c_m$. Consider the conjunctive view definition and view instance:

$$\begin{array}{lll} v_1(X,Y,Z) & :- & p(X,Y,Z) \\ v_2(X) & :- & r(X,Y) \\ v_3(Y) & :- & p(X,Y,Z),\ r(X,Z) \end{array} \qquad \begin{array}{rl} v_1 = & \{\langle i,j,1\rangle \mid x_i \text{ occurs in } c_j\} \\ & \cup\ \{\langle i,j,0\rangle \mid \bar{x}_i \text{ occurs in } c_j\} \\ v_2 = & \{\langle 1\rangle, \ldots, \langle n\rangle\} \\ v_3 = & \{\langle 1\rangle, \ldots, \langle m\rangle\} \end{array}$$

and the query: $q(c) :- r(X,Y),\ r(X,Y'),\ Y \neq Y'$.

We can show that tuple $\langle c\rangle$ is a certain answer under OWA if and only if formula φ is *not* satisfiable. Because the problem of testing a 3CNF formula for satisfiability is NP-complete [Coo71], this implies the claim. □

By Theorem 4.2, we know that maximally-contained queries compute exactly the certain answers under OWA. Because evaluating datalog queries has polynomial data complexity [Var82], it follows that in general there are *no* datalog queries that are maximally-contained in a conjunctive query with inequality.

First order queries We saw that adding recursion to positive queries leaves the data complexity of the problem of computing certain answers under OWA still polynomial. On the other hand, adding negation to positive queries makes the problem undecidable, as the following theorem shows (for both open and closed world assumptions). The proof is by reduction of functional and inclusion dependencies implication, that was shown to be undecidable [CV85, Mit83].

Theorem 4.5 *For $\mathcal{Q} \in FO$, the problem of determining, given a view definition together with a view instance, whether a tuple is a certain answer under the open or closed world assumption is undecidable, even if the view instance is empty.*

4.2 Positive view definitions

In the previous section, we proved that adding inequalities to the query results in co-NP-completeness of the problem of computing certain answers under OWA. The following theorem shows that allowing disjunction in the view definition has the same effect on the data complexity. The same result was proved by van der Meyden in [vdM93] while studying indefinite databases. We include the theorem for the sake of completeness.

Theorem 4.6 *[vdM93] For $\mathcal{V} \subseteq PQ$, $\mathcal{Q} \in CQ$, the problem of determining, given a view instance, whether a tuple is a certain answer under OWA is co-NP-hard.*

4.3 Datalog view definitions

Theorem 3.1 established that the problem can be solved in co-NP for $\mathcal{V} \subseteq PQ^{\neq}$ and $\mathcal{Q} \in datalog^{\neq}$. Here we examine the effect on the complexity of the problem of computing certain answers if we allow datalog as view definition language. For positive queries, the problem stays in co-NP as was shown by van der Meyden in [vdM93]. However, Theorems 4.7 and 4.8 respectively establish that the problem becomes undecidable for conjunctive queries with inequality and datalog queries.

Inequalities In the case of conjunctive view definitions, adding inequalities to the query increased the complexity of the problem of computing certain answers under OWA from polynomial to co-NP. With datalog view definitions, adding inequalities to the query raises the problem from co-NP complexity to undecidability. In [vdM92], van der Meyden showed undecidability for the case of $\mathcal{V} \subseteq datalog$ and $\mathcal{Q} \in PQ^{\neq}$. The following theorem proves that the problem is already undecidable for *conjunctive* queries with inequality.

Theorem 4.7 *For $\mathcal{V} \subseteq datalog$, $\mathcal{Q} \in CQ^{\neq}$, the problem of determining, given a view instance, whether a tuple is a certain answer under OWA is undecidable.*

Theorem 4.7 has an interesting consequence for the containment problem of a recursive datalog query in a nonrecursive datalog query with inequality. It shows that the technique in [CV92] to prove decidability of a datalog query in a nonrecursive datalog query does not carry to datalog with inequality. Indeed, it is an easy corollary of Theorems 4.1 and 4.7 that the problem is undecidable.

Datalog queries As we saw, there is a close relationship between the problem of computing certain answers under OWA and query containment. Not surprisingly it is therefore the case that the problem becomes undecidable for datalog view definitions and datalog queries.

Theorem 4.8 *For $\mathcal{V} \subseteq datalog$, $\mathcal{Q} \in datalog$, the problem of determining, given a view instance, whether a tuple is a certain answer under OWA is undecidable.*

4.4 First order view definitions

Theorem 4.5 showed that adding negation in queries leads to undecidability. The following theorem now shows that the same is true for adding negation to view definitions. (The proof is by reduction of first-order formula satisfiability that is undecidable.)

Theorem 4.9 *For $\mathcal{V} \in FO$, $\mathcal{Q} \in CQ$, the problem of determining, given a view instance, whether a tuple is a certain answer under the open or the closed world assumption is undecidable.*

5 Closed world assumption

Figure 1 (b) gives an overview of the complexity of computing certain answers under CWA. Computing certain answers under CWA is harder than computing certain answers under OWA. Whereas the problem is polynomial for $\mathcal{V} \subseteq CQ^{\neq}$ and $\mathcal{Q} \in datalog$ under OWA, the problem is already co-NP-complete for $\mathcal{V} \subseteq CQ$ and $\mathcal{Q} \in CQ$ under CWA. Moreover, whereas the problem is decidable for $\mathcal{V} \subseteq datalog$ and $\mathcal{Q} \in PQ$ under OWA, the problem is already undecidable for $\mathcal{V} \subseteq datalog$ and $\mathcal{Q} \in CQ$ under CWA.

5.1 Conjunctive view definitions

As in Theorem 4.6, the proof is a reduction of the problem of 3-colorability of a graph to the problem of computing certain answers.

Theorem 5.1 *For $\mathcal{V} \subseteq CQ$, $\mathcal{Q} \in CQ$, the problem of determining, given a view instance, whether a tuple is a certain answer under CWA is co-NP-hard.*

5.2 Datalog view definitions

The following result is proven using a straightforward reduction from the problem of containment between datalog queries, that is known to be undecidable [Shm93].

Theorem 5.2 *For $\mathcal{V} \subseteq datalog$, $\mathcal{Q} \in CQ$ the problem of determining, given a view instance, whether t is a certain answer under CWA is undecidable.*

6 View self-maintainability and view consistency

In this section, we consider two other important problems on materialized views, view consistency and view self-maintainability. We do it in the context of CWA since both of these problems make more sense in that context than under OWA.

Definition 6.1 (view consistency) Let $\mathcal{V}$ be a view definition and I an instance of the view. Then the view is *consistent* if there is a database D such that $I = \mathcal{V}(D)$. □

Definition 6.2 (view self-maintainability) Let D be a database. An *update* to D is either $d(R(t))$ (a deletion) for t a tuple in $D(R)$ or $i(R(t))$ (an insertion) for some tuple t not in $D(R)$. Let $\mathcal{V}$ be a view definition and I a consistent view instance. Then the view is *self-maintainable* for an update α if there exists a view instance J such that for each D with $I = \mathcal{V}(D)$, $J = \mathcal{V}(\alpha(D))$. □

views	self-maintainability	consistency
CQ	CO-NP	NP
$CQ^{\neq}$	CO-NP	NP
PQ	CO-NP	NP
datalog	undec.	undec.
FO	undec.	undec.

Figure 2: Complexity of the view self-maintainability and the view consistency problem

The complexity of these problems is shown in Figure 2. The complexity table for self-maintainability is the same as the one for computing certain answers under CWA in Figure 1 (b) for conjunctive queries. The complexity of the view consistency problem is similar with NP in place of co-NP. (The undecidable cases are r.e., whereas for computing certain answers and self-maintainability, they are co-r.e.)

Theorem 6.1

(i) For $\mathcal{V} \subseteq PQ^{\neq}$, the consistency is in NP *and the self-maintainability problem in co-*NP *(w.r.t. the size of the view).*

(ii) For $\mathcal{V} \subseteq CQ$, the consistency is NP*-hard and the self-maintainability problem in co-*NP*-hard (w.r.t. the size of the view).*

(iii) For $\mathcal{V} \subseteq datalog$ or $\mathcal{V} \subseteq FO$, the consistency problem is undecidable (r.e.) and the self-maintainability problem as well (co-r.e.).

7 Conclusion

We presented some complexity results with respect to materialized views. A main contribution is (i) the exhibition of deep connections with incomplete databases and (as a consequence) (ii) the point of view that a materialized view should be seen as an incomplete database. This indeed suggests using some model of incomplete information as the view model. We will illustrate briefly this direction with an example. Consider the self-maintainability problem of materialized views. Suppose we have such a view, the database is unavailable and we receive some updates to the database. A known technique is to verify whether the view is self-maintainable. If it is not, we raise our hands and in principle the view becomes unavailable. However, one could consider updating the incomplete database corresponding to the view. We could continue answering queries, and indeed, with such a model, it is possible to have more semantics in our answers, e.g. provide besides *certain* answers, *possible* answers, or indicate whether our answer is *surely* complete or not. We intend to continue the present work in that direction.

Acknowledgments We would like to thank Moshe Y. Vardi and Alon Y. Levy for pointing us to the work of Ron van der Meyden on the complexity of querying indefinite databases, and Pierre Huyn, Werner Nutt, Anand Rajaraman, and Yehoshua Sagiv for discussions on the topic.

References

[AD97] S. Abiteboul, O. M. Duschka. Complexity of answering queries using materialized views. Technical report, Stanford University, 1997. Available at URL http://logic.stanford.edu/people/duschka/papers/complexity.ps.

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

[AKG91] S. Abiteboul, P. Kanellakis, G. Grahne. On the representation and querying of sets of possible worlds. *TCS*, 78(1):159 – 187, 1991.

[ASU79a] A. V. Aho, Y. Sagiv, J. D. Ullman. Efficient optimiziation of a class of relational expressions. *ACM TODS*, 4(4):435–454, 1979.

[ASU79b] A. V. Aho, Y. Sagiv, J. D. Ullman. Equivalences among relational expressions. *SIAM J. on Computing*, 8(3):218–246, 1979.

[BLR97] C. Beeri, A. Y. Levy, M.-C. Rousset. Rewriting queries using views in description logics. In *PODS-97*, pp. 99 – 108, 1997.

[CKPS95] S. Chaudhuri, R. Krishnamurthy, S. Potamianos, K. Shim. Optimizing queries with materialized views. In *ICDE-95*, IEEE Comput. Soc. Press, pp. 190–200.

[CM77] A. K. Chandra, P. M. Merlin. Optimal implementation of conjunctive queries in relational data bases. In *ACM STOC*, pp. 77–90, 1977.

[Coo71] S. A. Cook. The complexity of theorem proving procedures. In *ACM STOCS*, pp. 151 – 158, 1971.

[CV85] A. K. Chandra, M. Y Vardi. The implication problem for functional and inclusion dependencies is undecidable. *SIAM J. on Computing*, 14(3):671–677, 1985.

[CV92] S. Chaudhuri, M. Y. Vardi. On the equivalence of recursive and nonrecursive datalog programs. In *PODS-92*, pp. 55–66.

[DG97a] O. M. Duschka, M. R. Genesereth. Answering recursive queries using views. In *PODS-97*, pp. 109 – 116.

[DG97b] O. M. Duschka, M. R. Genesereth. Query planning in Infomaster. In *ACM SAC*, 1997.

[DL97] O. M. Duschka, A. Y. Levy. Recursive plans for information gathering. In *IJCAI-97*.

[DS96] G. Dong, J. Su. Conjunctive query containment with respect to views and constraints. *IPL*, 57(2):95–102, 1996.

[Dus97] O. M. Duschka. Query optimization using local completeness. In *AAAI-97*, pp. 249–255.

[End72] H. B. Enderton. *A mathematical introduction to logic*. Academic Press, Inc., 1972.

[FV86] R. Fagin, M. Y. Vardi. The theory of data dependencies: A survey. In M. Anshel and W. Gewirtz, editors, *Mathematics of Information Processing: Proceedings of Symposia in Applied Mathematics*, vol. 34, pp. 19 – 71, 1986.

[Gra91] G. Grahne. *Problem of incomplete information in relational databases*. Springer-Verlag, 1991.

[Huy97] N. Huyn. Maintaining data warehouses under limited source access. Ph.D. Thesis STAN-CS-TR-97-1595, Stanford University, 1997.

[IJ84] T. Imielinski, W. Lipski Jr. Incomplete information in relational databases. *J. ACM*, 31(4):761–791, 1984.

[JK84] D. S. Johnson, A. Klug. Testing containment of conjunctive queries under functional and inclusion dependencies. *JCSS*, 28:167–189, 1984.

[Klu88] A. Klug. On conjunctive queries containing inequalities. *J. ACM*, 35(1):146–160, 1988.

[LMSS95] A. Y. Levy, A. O. Mendelzon, D. Srivastava, Y. Sagiv. Answering queries using views. In *PODS-95*.

[LRO96] A. Y. Levy, A. Rajaraman, J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *VLDB-96*, pp. 251–262.

[LRU96] A. Y. Levy, A. Rajaraman, J. D. Ullman. Answering queries using limited external processors. In *PODS-96*.

[LS97] A. Y. Levy, D. Suciu. Deciding containment for queries with complex objects. In *PODS-97*, pp. 32 – 43.

[LZW+97] W. J. Labio, Y. Zhuge, J. L. Wiener, H. Gupta, H. Garcia-Molina, J. Widom. The WHIPS prototype for data warehouse creation and maintenance. In *SIGMOD Conf.*, pp. 557 – 559, 1997.

[Mit83] J. C. Mitchell. The implication problem for functional and inclusion dependencies. *Information and Control*, 56(3):154 – 173, 1983.

[RSU95] A. Rajaraman, Y. Sagiv, J. D. Ullman. Answering queries using templates with binding patterns. In *PODS-95*.

[Shm87] O. Shmueli. Decidability and expressiveness aspects of logic queries. In *PODS-87*, pp. 237 – 249.

[Shm93] O. Shmueli. Equivalence of datalog queries is undecidable. *J. Logic Programming*, 15:231–241, 1993.

[SY80] Y. Sagiv, M. Yannakakis. Equivalence among relational expressions with the union and difference operators. *J. ACM*, 27(4):633–655, 1980.

[Ull88] J. D. Ullman. *Principles of Database and Knowledgebase Systems*, vol. 1. Computer Science Press, 1988.

[Ull89] J, D. Ullman. *Principles of Database and Knowledgebase Systems*, vol. 2. Computer Science Press, 1989.

[Ull97] J. D. Ullman. Information integration using logical views. In *ICDT-97*.

[Var82] M. Y. Vardi. The complexity of relational query languages. In *STOC-82*, pp. 137 – 146.

[Var87] M. Y. Vardi. Fundamentals of dependency theory. In E. Borger, editor, *Trends in Theoretical Computer Science*, pp. 171 – 224. Computer Science Press, 1987.

[vdM92] R. van der Meyden. The complexity of querying indefinite information: Defined relations, recursion and linear order. Technical report, Rutgers University, 1992.

[vdM93] R. van der Meyden. Recursively indefinite databases. *TCS*, 116(1):151–194, 1993.

[vdM97] R. van der Meyden. The complexity of querying indefinite data about linearly ordered domains. *JCSS*, 54(1):113 – 135, 1997.

[YL87] H. Z. Yang, P.-Å. Larson. Query transformation for PSJ-queries. In *VLDB-87*, pp. 245–254.