Partial Answers in Information Integration Systems

Gösta Grahne
Department of Computer Science
Concordia University
Montreal, Quebec, Canada
grahne@cs.concordia.ca

Victoria Kiricenko
Department of Computer Science
Concordia University
Montreal, Quebec, Canada
kiricen@cs.concordia.ca

Categories and Subject Descriptors

F.3 [**Theory of Computation**]: Logics and Meanings of Programs; H.2.1 [**Information Systems**]: Database Management—*Logical Design*; H.2.5 [**Information Systems**]: Database Management—*Heterogeneous Databases*

General Terms

Algorithms, Theory

Keywords

Information Integration, Query Rewriting Using Views, Incomplete Information

1. INTRODUCTION

Integrating heterogeneous data sources is a fundamental problem in databases, which has been studied extensively in the last two decades both from a formal and from a practical point of view (see e. g. [10, 4, 6]). One particular and useful way of viewing these systems, first proposed within the Information Manifold project [7], is to postulate a *global schema* (called a world view) that provides a unifying data model for all the information sources. A query processor is in charge of accepting queries written in terms of this global schema, translating them to queries on the appropriate sources, and assembling the answers into a global answer. Each source is modeled as a *materialized view* defined in terms of the global relations, which are virtual. State of the art query answering algorithms in these source–centric, so called local–as–view, information integration systems all produce a reformulated query that retrieves what has been thought of as "the best obtainable" answer, given the circumstances that the source–centric approach introduces incomplete information into the virtual global relations. However, perhaps since the relationship between information integration and incomplete information had not been clearly articulated, this "best obtainable" answer does not allow partial information. To illustrate the problem and to make this discussion more concrete let us consider a simple example.

Suppose the global schema contains two relations: *Prof(P, E, O, A)*, that is professor's name, email, office number, and research area. *Dept(P, D)*, that is professor's name and department.

The available sources are $\{S_1, S_2, S_3\}$. These sources have the following definitions:

$$S_1(P, E, O) \leftarrow Prof(P, E, O, A)$$

$$S_2(P, A) \leftarrow Prof(P, E, O, A)$$

$$S_3(P, D) \leftarrow Dept(P, D)$$

Suppose the user issues the query

$$Q(P, E, O, A) \leftarrow Prof(P, E, O, A), Dept(P, compsci)$$

That is, the user is interested in obtaining all available information about professors in the *compsci* department.

Since the information integration system does not have a way to get tuples for the subgoal *Prof* it would produce an empty rewriting and, thus, an empty answer for the user.

Everybody who has ever queried integrated information, especially in the case of the World Wide Web, knows this situation very well. The next logical action of the user, who wants to get at least partial information, is to try to make his(her) query less restrictive. In our example, to get at least some information about the professors in the *compsci* department, the user has to modify the original query by projecting out some of the attributes of the relation *Prof.* Since the user in unaware of the internals or the system, (s)he has to try all possible combinations of the attributes and then manually assemble the final answer. From the point of view of the user, the system should take on this burden and compute the answer containing this partial information. This is feasible, since the query could be rewritten as the union of the following unsafe conjunctive queries.

$$Q(P, E, O, X) \leftarrow S_1(P, E, O), S_3(P, compsci)$$

$$Q(P, X, Y, A) \leftarrow S_2(P, A), S_3(P, compsci)$$

The unrestricted variables X and Y represent unknown values. The answer can then be presented to the user as a table with some values missing, for example as the table below. (The contents of the table will, obviously, depend on the data provided by the sources.)

WIDM'03, November 7–8, 2003, New Orleans, Louisiana, USA.
Copyright 2003 ACM 1-58113-725-7/03/0011 ...$5.00.

Pname	Email	Office	Area
Murphy	murphy@cs.uoft.edu	SF322	⊥
Smith	⊥	⊥	DB
Jones	jones@cs.concordia.ca	LB1022	⊥
Murphy	⊥	⊥	AI

However, producing such partial answers is not without its intricacies. In a web based setting, the number of tuples in the answer is typically is much larger than the user wants. In such a case the user, to reduce the size of the answer, would most likely want to make his(her) query more restrictive. Which means that an information integration system, in order to be of any practical use, has to provide the user with the ability to search within the answers. This poses the restriction on how the answer to the original query is to be computed. The semantics of the query should be compositional, namely, if the second query is applied to the results of the first one the answer should be the same as if the composition of the two queries were executed on the underlying data sources.

In this paper we solve both problems illustrated by the above example. We first define the semantics of partial facts and introduce the notion of an exact answer — that is an answer that includes partial facts. We then provide two methods for computing the exact answer, in such way that semantics of queries remain compositional.

The first algorithm is a generalization of the "inverse–rules" approach [1], and involves explicitly computing a syntactic representation of the set of global databases implicitly defined by the sources. The other algorithm is a generalization of the rewriting technique (see e.g. [7, 10, 9]). This algorithm reformulates a query in terms of the source relations, and, thus, avoids inverting the entire source collection.

Due to the space limitation all proofs are omitted, however, they can be found in the full version of the paper.

2. THE MODEL

We begin by introducing some notions and defining the concepts that are used throughout the paper.

Let **rel** be a countably infinite set $\{R, S, \ldots, R_1, S_1, \ldots\}$ of *relation names*, let **dom** be a countably infinite set of *constants*, and let **var** be a countably infinite set of *variables*. Constants will be denoted by lower case letters and variables by upper case letters. Associated with each relation name R is a positive integer $arity(R)$, which is the arity of R.

A *fact* over R is an expression of the form $R(a_1, \ldots, a_k)$, where $k = arity(R)$, and each a_i is in **dom**. Let $\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$ be a set of relation names. A finite set of relation names will sometimes also be called a *schema*. A *database* d over $\mathbf{R}$ is a finite set of facts, each fact being over some $R_i \in \mathbf{R}$.

As most of papers on information integration do, we will focus on the most practical class of queries, namely the conjunctive queries. For this we need the concept of an *atom*. An atom is like a fact, except that we allow *variables*, taken from and infinite set **var**, as well as constants. For instance, $R(a, X)$ is an atom, whereas $R(a, b)$ is a fact. The variables in the atoms are placeholders, and they stand for any domain value. Variables will be denoted by uppercase letters, such as X and Y.

A *conjunctive query* φ is an expression of the form

$$head(\varphi) \leftarrow body(\varphi),$$

where $body(\varphi)$ is a set of atoms over relation names in the database schema, and $head(\varphi)$ is an atom over an answer relation name not used in the database. We assume that all variables occurring in $head(\varphi)$ also occur in $body(\varphi)$, i. e. that the query φ is *safe*. The variables occurring in $head(\varphi)$ are the *distinguished* variables of the query, and all the others are *existential* variables.

A conjunctive query φ can be applied to a database d resulting in a set of facts

$$\varphi(d) = \{\sigma(head(\varphi)) : \sigma(body(\varphi)) \subseteq d \text{ for some valuation } \sigma\}.$$

A *valuation* σ, is formally a finite partial mapping from $\mathbf{var} \cup \mathbf{dom}$ to **dom** that is the identity on **dom**.

Source collections and global databases

A *source* S is a pair (φ, v), where φ is a conjunctive query and v is a finite set of facts over $head(\varphi)$. A *source collection* $\mathcal{S}$ is a finite set of sources.

Since the global relations are virtual, in the sense that they don't contain any data, we need to answer the question: what is the instance of the global schema that is implicitly represented by the sources? As it turns out, the global database is actually incomplete. In other words, there might be several (usually infinitely many) global databases that are consistent with the definition of, and the data in the sources.

EXAMPLE 1. *For a simple example, suppose we have two sources* $S_1 = (V_1(X,Y) \leftarrow R(X,Y), \{V_1(a,b)\})$ *and* $S_2 = (V_2(X) \leftarrow R(X,Y), \{V_2(c)\})$. *Then it is natural to think that any database that contains at least the facts R(a,b) and R(c, e), for some* $e \in \mathbf{dom}$, *is a possible global database for* $\mathcal{S} = \{S_1, S_2\}$.

Formally, a source collection $\mathcal{S}$ defines a set of possible databases, denoted $poss(\mathcal{S})$, as follows:

$$\begin{aligned} poss(\mathcal{S}) = \{d \text{ over } sch(\mathcal{S}) \quad &: \quad v_i \subseteq \varphi_i(d) \text{ for all sources} \\ & \qquad S_i = (\varphi_i, v_i) \text{ in } \mathcal{S}\}. \end{aligned}$$

Since $poss(\mathcal{S})$ is infinite, we will now consider the problem of finitely representing an infinite set of databases. For this we invoke the venerable tableau.

Sets of databases and tableaux

Tableaux [8] are intended to concisely and finitely represent a large or infinite set of possible instances.

Let $\mathbf{R} = \{R_1, R_2, \ldots, R_n\}$ be a set of relation names. A *tableau* T over $\mathbf{R}$ is a finite set of atoms over the R_i's. Note that the same variable might appear in several atoms in T.

A tableau T over schema $\mathbf{R}$ represents a set of databases over $\mathbf{R}$. This set is denoted $rep(T)$, and it is defined by

$$rep(T) = \{d : \text{ there is a valuation } \sigma \text{ such that } \sigma(T) \subseteq d\}.$$

The definition says that a database d is represented by a tableau T, if there is a valuation σ such that when all variables in T are replaced by their image under σ, the set of facts thus obtained is a subset of d.

Representing $poss(\mathcal{S})$ by a tableau

Now the infinite set $poss(\mathcal{S})$ can be conveniently represented by a tableau denoted $T(\mathcal{S})$, such that

$$rep(T(\mathcal{S})) = poss(\mathcal{S}).$$

Example 2. *Suppose, for example, that we have one source* $S = (V(X,Z) \leftarrow R(X,Y), S(Y,Z), \{V(a,b), V(c,d)\})$. *Then* $T(S) = \{R(a,Y_1), S(Y_1,b),\ R(c,Y_2), S(Y_2,d)\}$, *where* Y_1 *and* Y_2 *are fresh variables.*

The details of how to construct $T(\mathcal{S})$ can be found in [3, 2].

3. QUERY ANSWERING

First we have to address the question of semantics of the query in information integration system. Since the global database is incomplete, and therefore, represents a set of possible databases, it is natural to expect that the answer to a query is also a set of answers, one for each possible database. With this in mind we define the exact answer.

Let $\mathcal{S}$ be source collection, and φ a conjunctive query over the global schema. Now φ applied to $\mathcal{S}$ defines the *exact answer*:

$$\varphi(\mathcal{S}) = \{\varphi(d)\ :\ d \in \mathit{poss}(\mathcal{S})\}.$$

The problem of computing exact answer to a user query was not addressed in the literature except for a brief discussion in [3] and somewhat more extensive treatment in [2]. Instead, all of the proposed algorithms are aimed at computing the possible or, most commonly, the certain answer defined as $\varphi^*(\mathcal{S}) = \bigcup\{\varphi(d) : d \in \mathit{poss}(\mathcal{S})\}$ and $\varphi_*(\mathcal{S}) = \bigcap\{\varphi(d) : d \in \mathit{poss}(\mathcal{S})\}$, respectively.

Essentially, the exact answer is the most general answer, and, given the exact answer we can obtain both possible and certain answers.

3.1 Computing the exact answer from the tableau

Since we are able to construct a database template T representing all databases in $\mathit{poss}(\mathcal{S})$ it is natural to extend the standard query evaluation mechanism to operate on database templates.

Given a conjunctive query φ over $\mathbf{R}$, our evaluation $\widehat{\varphi}$ (which is basically the "naive evaluation" of [5]) is defined next. For this we need the concept of a substitution. A *substitution* is a valuation, except that we allow variables to be mapped into variables, not only constants. Thus, a substitution θ is a function from (a subset of) $\mathbf{dom} \cup \mathbf{var}$ to $\mathbf{dom} \cup \mathbf{var}$, keeping in mind that constants have to be mapped to themselves. Then

$$\widehat{\varphi}(T) = \{\theta(\mathit{head}(\varphi)) : \theta(\mathit{body}(\varphi)) \subseteq T \text{ for some } \theta\}.$$

Example 3. *Let* $\varphi = Q(X,Y,Z) \leftarrow R(X,Y), S(Y,Z)$ *and* $T = \{R(a,b), R(d,X), S(b,c), S(X,e),\ S(Y,f)\}$. *Then* $\widehat{\varphi}(T) = \{Q(a,b,c), Q(d,X,e)\}$.

The fundamental result that

$$\mathit{rep}(\widehat{\varphi}(T)) \approx \varphi(\mathit{rep}(T))$$

was established in [5, 11], and adapted to tableaux in [3, 2]. Note that, we use coinitiality, instead of equality. Let $\mathcal{X}$ and $\mathcal{Y}$ be two enumerable sets of possible databases. We say that $\mathcal{X}$ and $\mathcal{Y}$ are *coinitial* if they have the same $\subseteq$–minimal elements. Coinitiality is denoted $\mathcal{X} \approx \mathcal{Y}$. For a deeper treatment of coinitiality, see [5].

As a consequence we now have a method for computing an $\approx$–approximation of $\varphi(\mathcal{S})$.

$$\mathit{rep}(\widehat{\varphi}(T(\mathcal{S}))) \approx \varphi(\mathcal{S}).$$

In other words, first invert the source extensions through their definitions, then apply the $\widehat{\varphi}$–evaluation of the user query φ on the resulting tableau. The result of the evaluation is another tableau, which the user perceives as a relation with nulls. Given the exact answer that is obviously most informative of all answers, we can compute the possible answer and the certain answer: $\varphi_*(\mathcal{S}) = \cap\, \mathit{rep}(\widehat{\varphi}(T(\mathcal{S})))$, and $\varphi^*(\mathcal{S}) \approx \cup\, \mathit{rep}(\widehat{\varphi}(T(\mathcal{S})))$.

Furthermore, we can prove that $\widehat{\varphi}$ evaluation has compositional semantics.

Lemma 1. $\mathit{rep}(\widehat{\varphi}(\widehat{\psi}(T(\mathcal{S})))) \approx \varphi(\psi(\mathcal{S}))$.

In essence, we can use our extended evaluation to compute a query on the result of another query.

However, computing $\widehat{\varphi}(T(\mathcal{S}))$ might involve a lot of redundant work, since it amounts to constructing the tableau corresponding to the entire source collection, whereas the global relations that are in $\mathit{body}(\varphi)$ might be mentioned in only few source definitions. Moreover, the query might have selections and joins that could be computed directly at the sources. For those reasons, a vast majority research in information integration focuses on query rewriting rather then just query answering.

3.2 Computing the exact answer directly on the source collection

In an information integration system a query processor is in charge of reformulating queries written in terms of the global schema to queries on the appropriate sources. This process is known as *query rewriting*. To obtain the answer to a given query the rewriting is then evaluated on the underlying sources. Since the exact answer is actually a set of answer sets and can be represented by a tableau, any rewriting to compute the exact answer has to be more general than the usual notion of rewriting.

In [2] we broaden query containment as follows. Let φ_1 and φ_2 be conjunctive queries. A query φ_1 is said to be *p–contained* in φ_2, denoted $\varphi_1 \subseteq_p \varphi_2$, if and only if there exists a conjunctive query ϕ, where φ_1 is equivalent to $\pi_L(\phi)$ (π is relational projection), for some list L of columns in $\mathit{head}(\phi)$ taken in the original order, such that for all databases d, $\phi(d) \subseteq \varphi_2(d)$. P–containment mappings, defined next, can be used to test p–containment of conjunctive queries. A *p–containment mapping* from a conjunctive query φ_2 to a conjunctive query φ_1 is a mapping μ, from variables of φ_2 to variables and constants of φ_1, such that $\mu(\mathit{body}(\varphi_2)) \subseteq \mathit{body}(\varphi_1)$, and, for every variable X in $\mathit{head}(\varphi_1)$, there is a variable Y in $\mathit{head}(\varphi_2)$, such that $\mu(Y) = X$.

Example 4. *Consider* $\varphi_1 = Q_1(X) \leftarrow R(X,Y), S(Y,Y), T(Y,Z)$ *and* $\varphi_2 = Q_2(A,B) \leftarrow R(A,B), S(B,C)$. *There is a p–containment mapping* $\mu = \{A/X, B/Y, C/Y\}$ *from* φ_2 *to* φ_1.

The notion of rewriting can be extended to *p–rewriting*.

The *expansion* of a query φ, denoted φ^{exp}, is obtained from φ by replacing all the source atoms in φ with their definitions. Existential variables in a source definition are replaced by fresh variables in φ^{exp}.

Let $\mathcal{S}$ be a source collection and φ be a conjunctive query over the global schema. The query ψ is a *p–contained rewriting* of φ using $\mathcal{S}$ if $\psi^{exp} \subseteq_p \varphi$.

Since we generalized the notion of containment mapping to p–containment mapping it is only natural that any rewriting algorithms (such as the algorithms in [7, 9]), which are based on containment mappings, can be extended to produce the p–rewriting. In [2] be give a generalization of of the bucket algorithm [7], which we call the p–bucket algorithm. In the full version of this paper we also give a generalization of the MiniCon algorithm [9], which constructs the p–rewriting.

Now we can address the issue of computing exact answer directly on the source collection.

Let ψ be a p–contained rewriting of φ. We define the φ–evaluation of ψ, denoted ψ_φ, as follows.

$$\psi_\varphi(\mathcal{S}) = \{\sigma_\mu(\mathit{head}(\varphi)) : \sigma(\mathit{body}(\psi)) \subseteq \mathit{ext}(\mathcal{S})\},$$

where μ is a p–containment mapping from φ to ψ^{exp} and σ is a valuation.

Suppose head variable X of φ is is mapped by μ to a variable in the expansion of subgoal $V_i(X_1, \ldots, X_k)$, and $\mu(X)$ is the j:th variable in the definition of S_i. Then the extension σ_μ of σ, is defined by setting

$$\sigma_\mu(X) = \begin{cases} \sigma(\mu(X)), \text{ if } \mu(X) \text{ occurs in } \mathit{head}(\psi) \\ f_{i,j}(\sigma(X_1), \ldots, \sigma(X_k)), \text{otherwise} \end{cases}$$

In order to define $\widetilde{\varphi}(\mathcal{S})$, the evaluation of conjunctive queries with function symbols in the head, we also need an auxiliary function *replace* that when applied to a set of atoms with function terms, replaces each unique term with unique variable. Now we set

$$\widetilde{\varphi}(\mathcal{S}) = \mathit{replace}\left(\bigcup\{\psi_\varphi(\mathcal{S}) : \psi^{exp} \subseteq_p \varphi\}\right),$$

and state the following important result:

THEOREM 1. *$\widetilde{\varphi}(\mathcal{S}) = \widehat{\varphi}(T(\mathcal{S}))$ up to renaming of the variables.*

EXAMPLE 5. *For example, consider the source collection $\mathcal{S} = \{(V_1(X_1, X_3) \leftarrow R(X_1, X_2),\ S(X_2, X_3),\ \{V_1(a, c)\}),$ $(V_2(X_1) \leftarrow T(X_1, X_2),\ \{V_2(c)\})\}$, and let the query φ be $Q(X, Y, Y, Z, W) \leftarrow R(X, Y), S(Y, Z), T(Z, W)$. In this case there is one (minimal) p–contained rewriting of φ, namely $\psi = Q(X, Y) \leftarrow V_1(X, Y), V_2(Y)$. Applying ψ_φ gives us the atom $Q(a, f_{1,2}(a),\ f_{1,2}(a), c,\ f_{2,2}(c))$, and applying the replace function yields the answer $\{Q(a, Y, Y, c, W)\}$. Had we used the tableau method instead, we would have gotten $T(\mathcal{S}) = \{R(a, Y), S(Y, c),\ T(c, W)\}$. Then applying $\widetilde{\varphi}$ to $T(\mathcal{S})$ would have given $\{Q(a, Y, Y, c, W)\}$.*

Note that since $\widetilde{\varphi}(\mathcal{S})$ computes a tableau that is equivalent to $\widehat{\varphi}(T(\mathcal{S}))$ the result of this computation can be used for subsequent querying, which was not possible with the evaluation that we presented in our earlier work [2].

It should be noted that the p–bucket algorithm can be modified to encode into the queries the containment mappings μ needed in the evaluation (see the full version of the paper).

4. SUMMARY

In the context of local–as–view information integration system a source collection defines a set of possible databases, therefore, querying a source collection, at least conceptually, means applying the query to each possible database, obtaining a set of possible answers. With this in mind we introduced the notion of the exact answer, which can be represented as a relation containing null values, and we gave two methods for computing the exact answer.

The first method involves explicitly computing $T(\mathcal{S})$ — a syntactic representation of the set of global databases implicitly defined by the sources. The $\widehat{\varphi}$ evaluation is then used to compute the exact answer. The other method reformulates a query in terms of the source relations, and, thus, avoids inverting the entire source collection. In order to achieve this we generalized classical notion of query containment to p–containment and used this notion to reformulate the query as a union of p–contained conjunctive queries. Then we defined the $\widetilde{\varphi}$–evaluation which uses p–contained rewritings to compute the exact answer on the source collection.

We have developed an experimental system, which enables us to evaluate the performance of the proposed methods to compute the exact answer in practice. As the result of our experiments we have established that computation of the exact answer can be done efficiently for all practical situations including the large-scale systems. Moreover, by comparing our results to the results presented in [9], we have confirmed that shifting from the computation of the certain answer to the computation of the exact answer does not change the average running time of query rewriting algorithms.

5. REFERENCES

[1] O. M. Duschka and M. R. Genesereth. Answering recursive queries using views. In *16th ACM Symp. on Principles of Database System*, pages 109–116, 1997.

[2] G. G. and V. Kiricenko. Obtaining more answers from information integration systems. In *5th Int'l Workshop on the Web and Databases*, pages 67–76, 2002.

[3] G. Grahne and A. O. Mendelzon. Tableau techniques for querying information sources through global schemas. In *7th Int'l Conference on Database Theory*, pages 332–347, 1999.

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

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

[6] M. Lenzerini. Data integration: A theoretical perspective (invited tutorial). In *21st ACM Symp. on Principles of Database Systems*, pages 233–246, 2002.

[7] A. Y. Levy, A. Rajaraman, and J. J. Ordille. Querying heterogeneous information sources using source descriptions. In *22nd Int'l Conf. on Very Large Databases (VLDB)*, pages 251–262, Mumbai (Bombay), India, 1996.

[8] A. O. Mendelzon. Database states and their tableaux. *ACM Trans. on Databases Systems*, 9(2):264–282, 1984.

[9] R. Pottinger and A. Y. Levy. A scalable algorithm for answering queries using views. In *26th Int'l Conference on Very Large Databases*, pages 484–495, 2000.

[10] J. D. Ullman. Information integration using logical views. In *6th Int'l Conference on Database Theory*, pages 19–40, 1997.

[11] M. Y. Vardi. Querying logical databases. *J. of Computer and System Sciences*, 33:142–160, 1986.