A Sound and Complete Query Evaluation Algorithm for Relational Databases with Null Values

Li Yan Yuan and Ding-An Chiang
The Center for Advanced Computer Studies
University of Southwestern Louisiana
Lafayette, LA 70504-4330

Abstract

Reiter has proposed extended relational theory to formulate relational databases with null values and presented a query evaluation algorithm for such databases. However, due to indefinite information brought in by null values, Reiter's algorithm is sound but not complete. In this paper, we first propose an extended relation to represent indefinite information in relational databases. Then, we define an extended relational algebra for extended relations. Based on Reiter's extended relational theory, and our extended relations and the extended relational algebra, we present a sound and complete query evaluation algorithm for relational databases with null values.

1. Introduction

There have been numerous attempts to represent incomplete information in the relational data model since the very beginning of the relational database theory [Co79, IL84, Re86, Vas79]. The main problem in this context has been the null value, a special symbol with some unknown properties. Reiter has proposed an extended relational theory to specify logical semantics of null values in the context of first order logic [Re86]. Based on his formal theory of databases and the extended relational theory, he has given a logical definition of what constitutes an answer to a query for relational databases with null values, and proposed a generalization of the relational algebra and a query evaluation algorithm. Unfortunately, although his evaluation algorithm is sound (it returns only correct answers), it is not complete (it does not return all correct answers). From both the theoretical and practical points of view, we do need a sound and complete evaluation algorithm for relational databases with null values [Mi87].

Example 1.1. [Re86]. Let $R = \{A, B\}$ be a relational database scheme, τ, the domain of A, is $\{\alpha\}$, θ, the domain of B, is $\{a, b, c\}$, where c is a null value, i.e. we know $a \neq b$, but c may be equal to a, or b. And, let $\mathbf{r} = \{(\alpha, a), (\alpha, b)\}$ be a relation over R. Consider a logical query $Q = <x \in \tau | (\exists y)(y \in \theta \wedge (x, y) \in \mathbf{r} \wedge y \neq c)>$. The answer to Q, denoted as $||Q||$, is $\{\alpha\}$, since we have $a \neq c \vee b \neq c$.

Reiter's algorithm returns an empty set for this query. We have observed that incompleteness of his algorithm for this example is due to ignoring indefinite information. Since $\{a \neq b\} \vdash a \neq c \vee b \neq c$, there exists an indefinite tuple $(\alpha, a) \vee (\alpha, b) \in \{(x, y) | (x, y) \in \mathbf{r} \wedge y \neq c\}$. Ignoring such indefinite information is one of the sources of Reiter's incompleteness. In order to obtain a sound and complete set of answers to relational databases with null values, we have to consider indefinite (disjunctive) information in relational databases, as shown by the above example.

Indefinite information has been considered in the context of deductive databases [GMN84]. However, representing indefinite information in relational databases is still a troublesome problem. Imielinski and Lipski have proposed the three representation systems to handle incomplete information in relational databases [IL84], and showed that such a system should be safe in the sense that the corresponding evaluation algorithm is sound and complete. However, their first two systems safely support projection, union, and positive selection, but not negative selection. Thus, they faced the same incompleteness problem as Reiter did. Their third system, as indicated by themselves, is mainly theoretical interest due to computing complexity.

In this paper, we first propose extended relations to handle indefinite information in relational databases. Informally, an extended relation is a set of sets of tuples over its scheme, i.e. each tuple of the relation is a set of tuples, which represent indefinite (disjunctive) information. If each set of tuples contains only one tuple, then the extended relation reduces to conventional relation. Then, we define extended relational algebra for extended relations. We show that all the extended relational algebra operators are sound in the sense that no incorrect conclusion is derived by using such operators. We observe that two operators, namely, intersection and

union, are not complete in the sense that not all valid conclusions are derivable. However, we show that the intersection is complete for definite tuples while union is complete for indefinite tuples. Thus, by distinguishing definite and indefinite tuples in extended relations, we resolve such incomplete problem, as shown in section 4. Therefore, we provide a safe relational algebra for extended relations with indefinite information. Another advantage of our extended relations and extended relational algebra for indefinite information is that they are based on relational theory and conform to the relational algebra, so they may be easily incorporated into an existing system.

Based on Reiter's extended relational theory and our extended relations, we present a sound and complete query evaluation algorithm for relational databases with null values, thus solving the problem raised by Reiter [Re86].

Reiter [Re86] is only concerned with definite answers in his evaluation algorithm, however, our algorithm returns not just definite answers, but all definite and indefinite answers to a given query. Due to difficulty of computing indefinite answers in primitively negative queries, the complexity of our algorithm is higher than his. As indicated by Vardi [Var85], query evaluation algorithms for databases with null values are more expensive than evaluation algorithms for databases without null values. We pay computing complexity to obtain all the correct results to queries in an incomplete database.

Reiter has considered two special cases, namely, positive queries and universal conjunctive queries, for which his algorithm is complete. We show that under the extended relational theory, the answer to any positive query contains only definite tuples, thus, our algorithm reduces to his and both algorithms have the same complexity. Furthermore, as far as only definite answers are concerned, our algorithm is sound and complete for existential quantifier free queries, which are more general than universal conjunctive queries, without invoking the difficulty of computing indefinite answers.

The rest of paper is organized as follows. In section 2, we review some preliminary results and give the notations that are used in the paper. In section 3, we propose extended relations and an extended relational algebra. Section 4 discusses properties of the extended relational algebra and present a sound and complete query evaluation algorithm for relational databases with null values. In section 5, some special cases are studied. Proofs of theorems in this paper can be found in [YC87].

2. Preliminaries

In this section, we briefly present some preliminary results and notations which are necessary for the discussion in the following sections. Reiter has proposed an extended relational theory to include the null value, an existing but unknown individual, using first-order logic [Re86]. We follow his extended relational theory here, thus the reader is suggested to read [Re86], if further explanation is needed.

2.1. Relational Language A *relational language* [Re86] is a first-order language (ALPHA, WFFS), where ALPHA is an alphabet of symbols and WFFS is the set of well-formed formulas(wffs) constructed from the symbols of ALPHA in the usual way. In addition, ALPHA has the following properties :

(1) Among the predicates of ALPHA there is a distinguished binary predicate E that is used as equality.

(2) Among the predicates of ALPHA, there is a distinguished nonempty subset of unary predicates, so called simple types, which model the concept of the domain of a relation attribute [Ul82].

In a relational language(ALPHA, WFFS), a term is a variable or a constant of ALPHA. If P is an n-ary predicate symbol of ALPHA, and t_1 ,..., t_n are terms, then $P(t_1, \ldots, t_n)$ is an atomic formula. For the convenience, we use the following abbreviations :

(1) If τ is a simple type, then
$(x/\tau)(W)$ abbreviates $(x)(\tau(x) \supset W)$, and
$(\exists x/\tau)(W)$ abbreviates $(\exists x)(\tau(x) \wedge W)$.
These two types of wffs are called type-restricted quantifier wffs, which are meant to restrict the possible x's to just those that belong to the class τ. Notice that quantifiers may be restricted only by types, not by arbitrary predicates.

(2) If $\mathbf{x} = x_1, \ldots, x_n$ is a sequence of distinct variables, then $W(\mathbf{x})$ abbreviates $W(x_1, \ldots, x_n)$, and $(\mathbf{x})W(\mathbf{x})$ abbreviates $(x_1) \cdots (x_n)W(x_1, \ldots, x_n)$.

(3) Let E be the equality predicate and $\mathbf{s} = s_1, \ldots, s_n$, and $\mathbf{t} = t_1, \ldots, t_n$ be sequences of terms. Then,
$E(\mathbf{s},\mathbf{t})$ abbreviates $E(s_1,t_1) \wedge \ldots \wedge E(s_n,t_n)$, and
$\neg E(\mathbf{s},\mathbf{t})$ abbreviates $\neg E(s_1,t_1) \vee \ldots \vee \neg E(s_n,t_n)$.

2.2. Extended Relational Theory Let (ALPHA, WFFS) be a relational language. A finite subset R of WFFS is an *extended relational theory* [Re86] iff R satisfies the following conditions :

(1) For each n-ary predicate P of ALPHA distinct from E (but including the simple types), R contains exactly one formula of the form
$$(\mathbf{x})P(\mathbf{x}) \equiv E(\mathbf{x},\mathbf{c}^{(1)}) \vee \ldots \vee E(\mathbf{x},\mathbf{c}^{(r)})$$

where the $\mathbf{c}^{(i)}$ are n-tuples of constants of ALPHA. The case r = 0 is permitted, in which case the corresponding formula is $(\mathbf{x})\neg P(\mathbf{x})$. This formula is called the *extension axiom* of P in R.

(2) R contains 0 or more *unique name axioms* of the form $\neg E(c, c')$ for distinct constants c, c' of ALPHA.

(3) Nothing else in R.

The extended relational theory provides a formal approach for modeling a relational database with null values. Reiter's paper and this one are concerned with null values that denote unknown individuals about which certain properties are known. For example, we know that a supplier who supplies a part p_1 may be s_1 or s_2, or may be a new fresh name, but we do know it is not s_3. Such null values have been problematic for the database theory ever since it was first proposed [Bi81, Co79]. In the extended relational theory, Reiter has adopted the closed world assumption and the so called an indexed null value, i.e., each time a new null value is introduced into the theory, the null must be denoted by a fresh name, distinct from all other names of the theory. The only thing that distinguishes the null value ω from the ordinary constants A, B etc, is the absence of unique name axioms for ω. As in the above example, a new unknown supplier is denoted by a fresh name s_ω, since s_ω may be s_1, s_2, or may be a new supplier, but not be s_3. Thus, there are no unique name axioms $\neg E(s_1, s_\omega)$, $\neg E(s_2, s_\omega)$ in the theory, but $\neg E(s_3, s_\omega)$ is in the theory.

2.3. Queries and Their Answers. Following [Re86], we define a query for a relational language (ALPHA, WFFS) to be any expression of the form $<\mathbf{x}/\tau|W(\mathbf{x})>$, where $\mathbf{x}/\tau$ denotes $x_1/\tau_1,\ldots,x_n/\tau_n$, each x_i is a distinct variable of ALPHA, each τ_i is a simple type of ALPHA, and $W(\mathbf{x}) \in$ WFFS is a wff whose free variables are among $x_1,\ldots,x_n$ and whose quantifiers are all typed quantifiers. The case $n = 0$ is permitted, in which case the query has the form $<|W>$, where W has no free variables. Reiter has discussed such a special case [Re86], we are not going to discuss such a case in this paper, however our algorithm can be easily extended for such a case.

Reiter has interpretated $<\mathbf{x}/\tau|W(\mathbf{x})>$ as the set of all n-tuples $\mathbf{x} = (x_1, \ldots, x_n)$ such that each x_i satisfies the simple type τ_i and the database satisfies $W(\mathbf{x})$. We consider more general interpretation of the query, as the formal definition below.

Definition 2.1. Let $DB \subseteq$ WFFS be any set of closed wffs, $<\mathbf{x}/\tau|W(\mathbf{x})>$ be a query with respect to DB, $\mathbf{t} = \{t_1, \ldots, t_m\}$, where $m \geq 1$, $t_i = (c_{i1}, \ldots, c_{in})$ be an n-tuple of constants. $\mathbf{t}$ is an answer to the query $<\mathbf{x}/\tau|W(\mathbf{x})>$ with respect to DB iff

(1) $DB \vdash \tau_j(c_{ij})$, $i = 1, \ldots, m$, $j = 1, \ldots, n$, and

(2) $DB \vdash W(t_1) \vee \ldots \vee W(t_m)$, and
$DB \nvdash W(t_1) \vee \ldots \vee W(t_{i-1}) \vee W(t_{i+1}) \vee \ldots \vee W(t_m)$, for $i = 1 \cdots m$, where $\vdash$ stands for "derive" and $\nvdash$ stands for "does not derive" in general logic meaning.

By the definition, $\mathbf{t} = \{t_1, \ldots, t_m\}$ is an answer to $<\mathbf{x}/\tau|W(\mathbf{x})>$ only if $W(t_1) \vee \ldots \vee W(t_m)$ is a minimally derivable ground formula from the extended relational theory. When $|\mathbf{t}| = 1$, $\mathbf{t}$ is said to be definite, otherwise, it is indefinite. The set of such answers is denoted as $||<\mathbf{x}/\tau|W(\mathbf{x})>||$. Furthermore, to distinguish definite and indefinite answers, we use the following notations:
$||<\mathbf{x}/\tau|W(\mathbf{x})>||_D = \{\mathbf{t} | \mathbf{t} \in ||<\mathbf{x}/\tau|W(\mathbf{x})>|| \ \&|\mathbf{t}|=1\}$, and
$||<\mathbf{x}/\tau|W(\mathbf{x})>||_I = \{\mathbf{t} | \mathbf{t} \in ||<\mathbf{x}/\tau|W(\mathbf{x})>|| \ \& \ |\mathbf{t}|>1\}$.

A query evaluation algorithm is *sound* iff for any query Q, the algorithm returns a subset of $||Q||$. Such an algorithm is *complete* iff it returns a superset of $||Q||$.

The work in [Re86] is only concerned with $||Q||_D$, i.e. definite answers. We have observed that even when only $||Q||_D$ is concerned, we have to take care of indefinite answers in the subqueries. Otherwise, we are unable to obtain $||Q||_D$. Ignoring indefinite answers is one of the sources of incompleteness of Reiter's algorithm.

Example 2.1. Consider the database in Example 1.1 in the context of the extended relational theory. That is, R is the extended relational theory having two simple types τ and θ and a binary predicate P, where
$(x)\tau(x) \equiv E(x, \alpha)$,
$(x)\theta(x) \equiv E(x, a) \vee E(x, b) \vee E(x, c)$,
$(\mathbf{x})P(\mathbf{x}) \equiv E(\mathbf{x}, (\alpha, a)) \vee E(\mathbf{x}, (\alpha, b))$,
with a single unique name axiom $\neg E(a, b)$.
Following Reiter's notation, we may use $|\tau|$ to denote the extension of τ, etc. That is,
$|\tau| = \{\alpha\}$, $|\theta| = \{a, b, c\}$, $|P| = \{(\alpha, a), (\alpha, b)\}$.
Consider the query
$$Q = <x/\tau|(\exists y/\theta)\, P(x, y) \wedge \neg E(y, c)>.$$
Then, $||Q|| = \{\alpha\}$, since $R \vdash (\exists y)(y \in \theta) \wedge P(\alpha, y) \wedge \neg E(y, c)$. On the other hand, by Reiter's decomposition algorithm
$$||Q|| \supseteq \Pi_\tau(||<x/\tau, y/\theta| \ P(x, y) \wedge \neg E(y.c)>||_D)$$
$$= \Pi_\tau(||<x/\tau, y/\theta| \ P(x, y)>||_D \cap (|\tau| \times ||<y/\theta| \ \neg E(y, c)>||_D)).$$
Let $||Q_1|| = ||<x/\tau, y/\theta| \ P(x, y)>||$,
$||Q_2|| = ||<y/\theta| \ \neg E(y, c)>||$.
Then, $||Q_1|| = ||Q_1||_D$.
But, $||Q_2||_D = \phi$, and $||Q_2||_I = \{\{a, b\}\}$.

That is, $||Q_2|| = \{ \{a, b\} \}$.
If we consider both definite and indefinite answers, and use extended relations and relational algebra for such situations, Then we are able to change his inclusion to equivalence to return all valid answers.
That is, $||Q||_D = \Pi_A(||Q_1||_D \cap (|\tau| \times ||Q_2||) = \{\alpha\}$.

2.4. More Notations Now, we define different types of queries for the following discussion.
A *positive* wffs is defined as follows [Re86]:

(1) An atomic wff is positive.

(2) If K is a positive wffs and τ a simple type, then $(x/\tau)K$ and $(\exists x/\tau)K$ are positive.

(3) If K_1 and K_2 are positive wffs, then so also are $K_1 \vee K_2$ and $K_1 \wedge K_2$.

(4) A wff is positive only by virtue of (1),(2), and (3).

A query $<x/\tau|K>$ is positive iff K is positive.

A *primitive positive* query is a query of the form $<x/\tau|P(r)>$ where $P(r)$ is an atomic formula, and all of the variables of x occur in r. P may be the equality predicate [Re86].

Let r be a relation consisting of n-tuples over τ, i.e., $r = \{t_1, \ldots, t_n\}$. Then we may define a formula $(x)P_r(x) \equiv E(x, t_1) \vee \ldots \vee E(x, t_m)$, as the extension axiom for r. $P_r(x)$ is called *pseudo atomic formula*.

A *primitive negative* query is a query of the form $<x/\tau|\neg P(r)>$ where $(x)P(x)$ is an atomic formula, or pseudo atomic formula, and P may be the equality predicate.

Definition 2.2. Let R be an extended relational theory, $t = t_1, \ldots, t_n$ be an n-tuple over τ. The set of all possible equivalent tuples of t, denoted as $PE(t)$, is defined as follows :
$PE(t) = \{u|u = u_1, \ldots, u_n$ is an n-tuple over τ, and for each $i = 1 \cdots n$, $\neg E(t_i, u_i)$ is not in $R\}$.

Assume $s = s_1, \ldots, s_n$ is in $PE(t)$. Then the set of alternative tuples for t and s, denoted as
$AE(t,s) = \{u_1, \ldots, u_n | u_i \in \{t_i, s_i\}$ for $i = 1 \cdots n\}$.

The absence of an unique name axiom represents the unknown properties of null values. Thus, $PE(t)$ is the set of null tuples which may be equal to t, and $AE(t,s)$ is the set of null tuples which are equal to t and s when t and s are equal.

Example 2.2 Let $t = \{(\alpha,a)\}$ and $s = \{(\beta,b)\}$. Then $PN(s,t) = \{(\alpha,a),(\alpha,b)(\beta,a),(\beta,b)\}$.

3. Extending Relations with Indefinite Tuples

Ignoring indefinite tuples in relations is one of the sources of incompleteness in Reiter's theory. In this section, we first extend relations with indefinite (disjunctive) tuples, then define an extended relational algebra for such extension. In the next section, we will see that extending relations with indefinite tuples will give us a sound and complete query evaluation algorithm for relational databases with null values.

Yahya & Henschen have defined PGC(P), the set of all minimally derivable ground clauses for P in the context of deductive databases [YH85]. That is, for each predicate $P(x)$ in DB. PGC(P) = $\{K \mid K = c_1 \vee \ldots \vee c_n$ is a ground clause, and $DB \mid\!\!- P(c_1) \vee \ldots \vee P(c_n)$ and $DB \not\mid\!\!- P(c_1) \vee \ldots \vee P(c_{i-1}) \vee \ldots \vee P(c_{i+1}) \vee \ldots \vee P(c_n)$, for each $i = 1 \cdots n\}$.

To represent P in a relational database, we have to modify the conventional definition of a relation in order to store such disjunctive tuples.

Definition 3.1. Let $R = \{z_1, z_2, \ldots z_n\}$ be a relational scheme, where z_i is an attribute, and $D(z_i)$ be the domain of z_i. Then an extended relation r over R is defined as follows:
$r \subseteq \{t|t = \{t_1, \cdots, t_m\}$, where $m \geq 1$, and
$t_i \in D(z_1) \times \ldots \times D(z_n)$, for $i = 1 \cdots n\}$.
When for each $t \in r$, $|t| = 1$, then the extended relation r reduces to a conventional relation. In the relational theory, it is required that there is no duplicate tuples in any relation. Similarly, we define the notion of a redundant tuple for the extended relation.

Definition 3.2. Let r be an extended relation, t_1 be a tuple in r. t_1 is said to be *redundant* if there exists a tuple t_2 in r such that $t_2 \subseteq t_1$.

It is desirable that extended relations do not contain redundant tuples. We assume that extended relations do not contain redundant tuples. When they do, the system automatically eliminates redundant tuples.

For any query $Q = <x/\tau|W(x)>$ to a database DB, the answer $||Q||$ can be represented by an extended relation.

Now, we define some of the extended relational algebra operators, which are useful for the following discussions. Using the similar idea, it is not difficult to define other extended relational algebra operators.

Extended Relational Union. Let r_1 and r_2 be compatible extended relations. Then,
$r_1 \cup r_2 = \{t|t \in r_1$ or $t \in r_2\}$.
Example 3.1. Let $r_1 = \{ \{(a,b)\}, \{(c,d),(c,b)\} \}$, $r_2 = \{ \{(a,b)\}, \{(c,d)\}, \{(a,d),(b,e)\} \}$. Then, $r_1 \cup r_2 = \{\{(a,b)\}, \{(c,d)\}, \{(a,d),(b,e)\}\}$.†

† Notice that the result should not contain redundant tuples.

Extended Relational Intersection. Let r_1 and r_2 be compatible extended relations. Then,

$r_1 \cap r_2 = \{t |\ t \in r_1$ and for each $c \in t$, $c \in r_2$ or $t \in r_2$ and for each $c \in t$, $c \in r_1\}$.

Example 3.2. Let
$r_1 = \{\{(a,b)\}, \{(a,c),(b,d)\}, \{(e,f)\}\}$,
$r_2 = \{\{(a,b)\},\{(a,c)\},\{(b,d)\},\{(c,e),(c,d)\}\}$. Then,
$r_1 \cap r_2 = \{\{(a,b)\}, \{(a,c),(b,d)\}\}$.†

Extended Relational Selection. Let r be an extended relation, F be a selection formula as defined for the conventional selection in [Ul82]. Then,

$\sigma_F(r) = \{t |\ t \in r$ and $F(c_i)$ is true for each $c_i \in t\}$

Example 3.3. Let
$r = \{\{(a,b,a)\},\{(d,e,d),(c,f,c)\},\{(a,c,f),(a,c,a)\}\}$.
Then, $\sigma_{1=3}(r) = \{\{(a,b,a)\}, \{(d,e,d),(c,f,c)\}\}$.†

Extended Relational Projection. Let r be an extended relation over $R(x,y)$.

$\Pi_x(r) = \{t | \exists s \in r$ such that $t = \Pi_x'(s)\}$,

where $\Pi_x'(s)$ is the conventional relational projection [Ul82].

Example 3.4. Let
$r = \{\{(a,b)\}, \{(a,c),(b,d)\}, \{(e,f),(e,g),(c,g)\}\}$.
Then, $\Pi_1(r) = \{\{a\}, \{e,c\}\}$.†

Extended Cartesian Product. Let r_1 and r_2 be extended relations. Then,

$r_1 \times r_2$
$= \{t | \exists s_1 \in r_1, \exists s_2 \in r_2$ such that $t = r_1 \times' r_2\}$,

where $\times'$ is the conventional cartesian product [Ul82].

Example 3.5. Let $r_1 = \{\{(a,b),(c,d)\}\}$,
$r_2 = \{\{(e,f)\}, \{(e,g),(f,e)\}\}$. Then,
$r_1 \times r_2 = \{\{(a,b,e,f),(c,d,e,f)\}, \{(a,b,e,g),(a,b,f,e),$
$(c,d,e,g),(c,d,f,e)\}\}$.

Extended Relational Division. Let r be an extended relation with n attributes, θ be an extended relation such that for each $t \in \theta$, $|t| = 1$. Then,

$\Delta_\theta(r) = \{t | t = \{t_1, \ldots, t_n\}$ and $\forall \alpha \in \theta(\{t_1\alpha, \ldots, t_n\alpha\} \in r)\}$.

Example 3.6. Let $\theta = \{b\}$, and $r = \{\{(a,b)\}, \{(a,c)\}, \{(b,b),(c,b),(d,b)\}, \{(e,b),(e,f)\}\}$. Then,
$\Delta_\theta(r) = \{\{a\}, \{b,c,d\}\}$.†

Extended relations with the extended relational algebra defined above provide a powerful tool for storing indefinite information in relational databases. Since such an extension is based on the relational theory, it can be easily incorporated into an existing relational system.

† Notice that the result should not contain redundant tuples.

4. A Sound and Complete Evaluation Algorithm.

In this section, we propose a sound and complete evaluation algorithm for any given query against an extended relational theory R. The algorithm is designed to conform to the extended relational algebra discussed in the previous section. As with Reiter's algorithm, because of our conformity to the relational algebra, our algorithm could be incorporated into an existing relational system.

The algorithm proceeds by recursively decomposing complex queries in a certain normal form into suitable extended relational operations on simpler queries. Moreover, our algorithm returns not just the set of definite answers, but the set of all definite and indefinite answers. The complexity of the algorithm is relatively higher than Reiter's algorithm, however, as indicated by Vardi [Var85], the complexity of query evaluation in the presence of null values would be computationally more expensive than that in the absence of null values. Furthermore, as we shall see in the next section, our algorithm is sound and complete in a more general case than Reiter's algorithm did at the same complexity.

In Reiter's algorithm, the union and existential quantifier are two sources of incompleteness. We propose a method, as shown in theorem 4.2, to solve the problem of union. However, it is the indefinite information that causes the incompleteness in the existential quantifier query. Even when we only need to find a complete set of definite answers, we must deal with indefinite information, as indicated in Example 1.1 and 2.1.

Theorem 4.1. Let R be an extended relational theory, then

(1) $||<x/\tau|W_1(x) \wedge W_2(x)>|| \supseteq ||<x/\tau|W_1(x)>|| \cap ||<x/\tau|W_2(x)>||$.

(2) $||<x/\tau|W_1(x) \wedge W_2(x)>||_D = ||<x/\tau|W_1(x)>||_D \cap ||<x/\tau|W_2(x)>||_D$.

(3) $||<x/\tau|W_1(x) \wedge W_2(x)>|| = ||<x/\tau|W_1(x)>|| \cap ||<x/\tau|W_2(x)>||$, where W_1 or W_2 is positive.

When both $W_1(x)$ and $W_2(x)$ are not positive, (3) may not hold as shown in the following example.

Example 4.1. Consider the extended relational theory with a single type, τ; where $|\tau| = \{a,b,c,d,e,f\}$, and unique name axioms are $\neg E(a,b)$, $\neg E(a,c)$, $\neg E(a,d)$, $\neg E(b,c)$, $\neg E(b,d)$, $\neg E(e,d)$, and $\neg E(f,c)$.
Then, $||<z/\tau|\neg E(z,e) \wedge \neg E(z,f))>|| = \{\{a,b,c\}, \{a,b,d\}, \{a,c,d\}, \{b,c,d\}\}$. But,
$||<z/\tau|\neg E(z,e)>|| = \{\{d\},\{a,b\},\{a,c\},\{b,c\},\{c,f\}\}$,
and $||<z/\tau|\neg E(z,f)>|| = \{\{c\}, \{a,b\}, \{a,d\}, \{b,d\},\{d,e\}\}$.

Thus, $||<x/\tau|\neg E(x,c)>|| \cap ||<x/\tau|\neg E(x,f)>|| = \phi$.

Theorem 4.1 tells us that conjunction of two non-positive queries may cause some incompleteness as shown by the above example. The following theorem indicate similar situation for disjunction.

Theorem 4.2. Let R be an extended relational theory, then

(1) $||<x/\tau|W_1(x) \vee W_2(x)>||$
$\supseteq ||<x/\tau|W_1(x)>|| \cup ||<x/\tau|W_2(x)>||$.

(2) $||<x/\tau|W_1(x) \vee W_2(x)>||_I$
$= ||<x/\tau|W_1(x)>||_I \cup ||<x/\tau|W_2(x)>||_I$.

(3) Let $W_1(x)$ and $W_2(x)$ be positive wffs. Then,
$||<x/\tau|W_1(x) \vee \neg W_2(x)>||_D$
$= ||<x/\tau|W_1(x)>||_D \cup ||<x/\tau|\neg W_2(x)>||_D$
$\cup \{t|$for each $s \in (PE(t) \cap |W_2|)$, $AE(s,t) \cap |W_1| \neq \phi\}$.

Consider (3) in the above theorem, when W_2 is a positive wffs, we may write (3) as
$||<x/\tau|W_1(x) \vee \neg W_2(x)>||_D$
$= ||<x/\tau|W_1(x)>||_D \cup_E ||<x/\tau|\neg W_2(x)>||_D$.
The reverse inclusion of theorem 4.2.(1) does not hold, which is another source of incompleteness in Reiter's algorithm. However, using theorem 4.2.(3), we resolve this problem, as shown in the following example.

Example 4.2 [Re86]. Consider the theory with a single type τ, where $|\tau| = \{a,b\}$, a single predicate $(x)P(x)$, where $|P| = \{a\}$, and no unique name axioms. Then,
$||<x/\tau|P(x) \vee \neg P(x))>||$
$= ||<x/\tau|P(x)>|| \cup_E ||<x/\tau|\neg P(x))>|| = \{a,b\}$.
Since $PE(a) = \{a,b\}$, $PE(b) = \{a,b\}$, $AE(a,b) = \{a,b\}$, and $a \in |P|$ and $b \notin |P|$.

The next theorem allows us to strip off leading typed quantifiers in queries.

Theorem 4.3. Let R be an extended relational theory, and $W(x,y)$ be a (possibly quantified) formula with free valables among $x = x_1, \ldots, x_n$ and y. Then,

(1) if $|\theta|=\{\}$, then $||<x/\tau|(y/\theta)\ W(x,y)>|| = |\tau|$,

(2) if $|\theta|\neq\{\}$, then $||<x/\tau|(y/\theta)\ W(x,y)>||$
$=\Delta_\theta(||<x/\tau,y/\theta|\ W(x,y)>||)$,

(3) $||<x/\tau|(\exists y/\theta)W(x,y)>||$
$= \Pi_x(||<x/\tau,y/\theta|\ W(x,y)>||)$.

With the introduction of indefinite tuples in relational query answers, we resolve the existential quantifier problem, but we pay computing complexity for this.

Theorem 4.4. [Re86]. Suppose R is an extended relational theory, and the variable y does not occur free in the formula $W(x)$. Then,

(1) $||<y/\theta,x/\tau|\ W(x)>|| = |\theta| \times ||<x/\tau|\ W(x)>||$,

(2) if for $n \geq 1$, $x/\tau = x_1/\tau_1,\ldots,x_n/\tau_n$ and for $k \geq 0$ $z/\psi = z_1/\psi_1, \ldots, z_k/\psi_k$, then
$||<x/\tau,y/\theta,z/\psi|\ W(x,z)>|| = \Pi_{2,\ldots,n+1,1,n+2,\ldots,n+k}(|\theta| \times ||<x/\tau,z/\psi\ |W(x,z)>||)$.

Example 4.3. Consider R and Q in Example 2.1.
$||Q|| = \Pi_x(||<x/\tau,y/\theta|\ P(x,y) \wedge \neg\ E(y,c)\ >||)$
$= \Pi_x(||<x/\tau,y/\theta|\ P(x,y)>|| \cap$
$||<x/\tau,y/\theta|\ \neg\ E(y,c)\ >||\ )$
$= \Pi_x(\ \{(\alpha,a),(\alpha,b)\} \cap (\{\alpha\} \times \{\{a,b\}\})\ )$
$= \Pi_x(\ \{(\alpha,a),(\alpha,b)\} \cap \{\{(\alpha,a),(\alpha,b)\}\}\ )$
$= \Pi_x(\ \{\{(\alpha,a),(\alpha,b)\}\}\ ) = \{\alpha\}$.

Definition 4.1. Let $r = r_1, \ldots, r_n$ be an n-tuple of variables and/or constants, $x=x_1,\ldots,x_m$ be a sequence of distinct variables, where each x_i is a variable occurring in r. Then, $F(r,x)$ is a set of clauses obtained as follows :

(1) If r_i is a constant, then $i=r_i$ is in $F(r,x)$.

(2) If r_i is a variable, say x_k, and r_j, $1 \leq j \leq n$, is an occurrence of x_k. Then $i=j$ is in $F(r,x)$.

(3) Nothing else is in $F(r,x)$.

We may write $F(r,x)$ as a conjunction of clauses in $F(r,x)$.

Example 4.4. Let $r = x,y,a,x,z,y$ and $x = x,y,z$. Then, $F(r,x) \equiv 1=4 \wedge 2=6 \wedge 3=a$.

Theorem 4.5. Let R be an extended relational theory and $<x/\tau|P(r)>$ be a positive primitive query. Then, $||<x/\tau|P(r)>|| = |\tau| \cap \Pi_x(\sigma_{F(r,x)}|P|)$.

Theorem 4.6. Let R be an extended relational theory and $<x/\tau|\neg P(r)>$ be a negative primitive query, where r be an n-tuple of variables and/or constants over θ, t be a sequence of n-distinct variables, $P(t)$ be an extension axiom standing for the relation $||<t/\theta|P(t)>||$. Then,
$||<x/\tau|\neg P(r)>|| = |\tau| \cap \Pi_x(\sigma_{F(r,x)}(||<t/\theta|\neg P(t)>||))$.

Our remaining task for computing primitive query is to determine a negative query of the form $||<z/\tau|\neg P(z)>||$, where $P(z)$ is an atomic formula. The following theorem presents an algorithm to do so.

Definition 4.2. Let R be the extended relational theory, $(x/\tau)P(x)$ be the extension axiom for a relation $|P|$, and $t = \{t_1, \ldots, t_m\}$ be a set of tuples over τ. Then,
$NE_P(t) = \{\ x|x \in \tau \wedge x \in |P| \wedge x \in PE(t_i)$ for some $t_i \in t\}$.

Theorem 4.7. Let R be an extended relational theory,

$Q = \langle x/\tau | \neg P(x) \rangle$ be a negative primitive query, and $\mathbf{t} = \{t_1, \ldots, t_m\}$ be a set of tuples over τ. Then, $\mathbf{t} \in ||Q||$ iff

(1) For each $t_i, t_j \in \mathbf{t}$, either $\neg E(t_i, t_j)$ or $NE_p(t_i) \cap NE_p(t_j) = \phi$, and

(2) $|NE_p(\mathbf{t})| = |\mathbf{t}| - 1$, and

(3) there is no proper subset $\mathbf{t}'$ of $\mathbf{t}$ such that $|NE_p(\mathbf{t}')| = |\mathbf{t}'| - 1$.

Example 4.5. Let R be the extended relational theory with a single simple type τ, where $\tau = \{a, b, c, d, e\}$, and unique name axioms $\neg E(a,b)$, $\neg E(a,c)$, and $\neg E(b,c)$.
The query Q is $\langle x/\tau | \neg E(x,d) \wedge \neg E(x,e) \rangle$.
Then, we may transform Q into Q', where
$Q' = \langle x/\tau | \neg(E(x,d) \vee E(x,e)) \rangle$.
Let $P(t) = E(t,d) \vee E(t,e)$. That is,
$|P(t)| = ||\langle t/\tau | E(t,d) \vee E(t,e) \rangle|| = \{d, e\}$.
Thus $||Q|| = ||\langle x/\tau | \neg P(x) \rangle|| = \{\{a,b,c\}\}$.
Since $NE_p(\{a,b,c\}) = \{a,b,c,d,e\} \cap |P| = \{d,e\}$ and $|NE_P(\{a,b,c\})| = |\{a,b,c\}| - 1$.

Computing $||\langle x/\tau | \neg P(x) \rangle||$ may be time consuming. However, in order to obtain complete query answers, we have to confront this situation.

Now, we are in the position to present our sound and complete query evaluation algorithm for relational databases with null values. Given a query $\langle x/\tau | W(x) \rangle$, if there are some typed quantifiers in $W(x)$, then we can use the usual validity preserving transformations to transform $W(x)$ into the prenex normal form, that is, all quantifiers are leading quantifiers. Then by theorem 4.3, we need only consider those queries which are quantifier free.

The failure of the reverse inclusion in theorem 4.1(1) and 4.2(1) prevents us from randomly decomposing a quantifier free query into primitive queries. Fortunately, theorem 4.1 and 4.2 indicate that such failure applies only to definite and indefinite answers depending on different situations. If we try to obtain definite and indefinite answers separately, then we can find a sound and complete query evaluation algorithm.

Definition 4.3. Let W be a quantifier free wff, W is said to be in the pseudo normal form, if W is of the form $(\bigvee_{i=1}^{m}(W_{i1} \wedge \neg W_{i2})) \wedge W_{m+1}$,
where W_{m+1}, and W_{i1}, $1 \le i \le m$, are positive wffs, and W_{i2}, $1 \le i \le m$, are the disjunctions of atomic wffs.

For any quantifier free wff W, we may find the equivalent wff in the pseudo normal form as shown by the following example.

Example 4.6. Let $W = (W_{11} \vee \neg W_{12}) \wedge (W_{21} \vee \neg W_{22}) \wedge (W_{31} \vee W_{32})$, where each W_{ij} is an atomic formula. Then,
$W \equiv ((W_{11} \wedge W_{21}) \vee (W_{11} \wedge \neg W_{22}) \vee (W_{21} \wedge \neg W_{12}) \vee (\neg W_{12} \wedge \neg W_{22})) \wedge (W_{31} \vee W_{32})$
$\equiv ((W_{11} \wedge W_{21}) \vee (W_{11} \wedge \neg W_{22}) \vee (W_{21} \wedge \neg W_{12}) \vee \neg(W_{12} \vee W_{22})) \wedge (W_{31} \vee W_{32})$.
The latter is in the pseudo normal form.

Theorem 4.8. Let R be an extended relational theory, $\langle x/\tau | W(x) \rangle$ be a quantifier free query, $W_1 = \bigwedge_{j=1}^{n}(W_{j1} \vee \neg W_{j2} \vee \neg W_{j3} \vee \ldots \vee \neg W_{jl_j})$ be the equivalent conjunctive normal form of W, and $W_2 = (\bigvee_{i=1}^{m}(W_{i1} \wedge \neg W_{i2})) \wedge W_{m+1}$ be the equivalent pseudo normal form of W,
where W_{j1}, W_{i1}, W_{m+1} are positive wffs, W_{jk}, $2 \le k \le l_j$ are atomic wffs, W_{i2}, $1 \le i \le m$, are disjunctions of atomic wffs. Then,
$||Q_1|| = ||\langle x/\tau | W_1(x) \rangle||_D$
$= \bigcap_{j=1}^{n}(\bigcup_{k=1}^{l_j} E(||\langle x/\tau | W_{jk}(x) \rangle||_D))$,
$||Q_2|| = (\bigcup_{i=1}^{m}(||\langle x/\tau | W_{i1}(x) \rangle|| \cap ||\langle x/\tau | \neg W_{i2}(x) \rangle||)) \cap ||\langle x/\tau | W_{m+1}(x) \rangle||$
$\supseteq ||\langle x/\tau | W_2(x) \rangle||_I$, and
$||\langle x/\tau | W(x) \rangle|| = ||Q_1|| \cup ||Q_2||$.

Example 4.7. Let R be an extended relational theory having a single simple type τ, three predicates, and unique name axioms as shown below,
$|\tau| = \{a,b,c,d,e\}$, $|P_1| = \{a\}$, $|P_2| = \{a,e\}$,
$|P_3| = \{e\}$, and $\neg E(a,c)$, $\neg E(a,d)$, $\neg E(c,d)$, and $\neg E(b,e)$. Consider a query
$Q = \langle x/\tau | (P_1(x) \vee \neg P_2(x)) \wedge \neg P_3(x) \rangle$. Then,
$W_1 = (P_1(x) \vee \neg P_2(x)) \wedge \neg P_3(x)$ is in the conjunctive normal form, while
$W_2 = (P_1(x) \wedge \neg P_3(x)) \vee \neg(P_2(x) \vee P_3(x))$ is in the pseudo normal form. Thus,
$||Q_1|| = ||Q||_D$
$= (||\langle x/\tau | P_1(x) \rangle||_D \cup_E ||\langle x/\tau | \neg P_2(x) \rangle||_D) \cap ||\langle x/\tau | \neg P_3(x) \rangle||_D$
$= \{a,b\} \cap \{b\} = \{b\}$.
$||Q_2|| = (||\langle x/\tau | P_1(x) \rangle|| \cap ||\langle x/\tau | \neg P_3(x) \rangle||) \cup (||\langle x/\tau | \neg(P_2(x) \vee P_3(x)) \rangle||)$
$= (||\langle x/\tau | P_1(x) \rangle|| \cap ||\langle x/\tau | \neg P_3(x) \rangle||) \cup ||\langle x/\tau | \neg P_4(x) \rangle||$
$= (\{\{a\}\} \cap \{\{b\}, \{a,c\}, \{a,d\}, \{c,d\}\}) \cup \{\{c,d\}\} = \{\{c,d\}\} \supseteq ||Q_2||_I$,
where $|P_4| = |P_2| \cup |P_3| = |P_2|$.
Thus, $||Q|| = ||Q_1||_D \cup ||Q_2|| = \{\{b\}, \{c,d\}\}$.

In the above example, we cannot obtain a sound and complete set of answers without considering definite and indefinite answers separately. In fact, we may

obtain a sound and complete answer by any decomposition of a given query, as long as we avoid intersection and/or union of two nonpositive wffs, which are sources of incompleteness, as indicated in the theorem 4.1(1) and 4.2(1).

5. Some Special Cases

Reiter has shown that there are three special cases for which his decomposition algorithm is not only sound but also complete for relational databases: Universally quantified conjunctive queries, positive queries, and databases without null values. Since his algorithm deals with only definite answers, so it is more efficiency than ours.

In this section, we discuss the special cases in order to compare two evaluation algorithms. For the last two special cases, our algorithm reduces to his as shown below. Furthermore, as long as only definite answers are required, our algorithm returns the set of all definite answers without considering any indefinite answers for existential quantifier free queries, which is a more general case than his first special case. Since we do not consider indefinite answers, both algorithms are of the same complexity.

5.1 Positive queries. If the query is positive, then, as shown by the following theorem, there is no indefinite answers, so our algorithm reduces to his.

Theorem 5.1. Let $||<x/\tau|W(x)>||$ be a positive query. Then, $||<x/\tau|W(x)>|| = ||<x/\tau|W(x)>||_D$.

5.2 Relational databases without null values. An extended relational theory is said to be a relational theory iff for each pair of distinct constants c, c', $\neg E(c,c') \in R$ or $\neg E(c',c) \in R$. Thus, a relational database without null values can be represented by a relational theory. The following theorem shows that in this case, there are no indefinite answers, and theorem 4.2(1) can be reversed.

Theorem 5.2. Let R be a relational theory. Then

(1) $||<x/\tau|W(x)>|| = ||<x/\tau|W(x)>||_D$,

(2) $||<x/\tau|W_1(x) \vee W_2(x)>||$
$= ||<x/\tau|W_1(x)>|| \cup ||<x/\tau|W_2(x)>||$.

5.3 Existential quantifier free queries. An existential quantifier free query is a query without existential quantifiers. Thus, it is more general than universally quantified conjunctive queries, since it may contain disjunctions.

The failure of the reverse of inclusion in theorem 4.2.(1) prevents Reiter's algorithm from obtaining a sound and complete set of all definite answers. However, with the contribution of theorem 4.2.(3), we are able to reverse such an inclusion. Therefore, our algorithm returns the set of all definite answers without computing any indefinite answers. Thus, our algorithm is of the same complexity as Reiter's, but is sound and complete in the more general case.

REFERENCES

[Bi81] Biskup, J. "Null Values in Database Relations", In *Advances in Data Base Theory*, vol. 1, H. Gallaire, J. Minker, and J. M., Nicolas, Eds. Plenum Press, New York, 1981, pp. 299-341.

[Co79] Codd, E, F. "Extending the Database Relational Model to Capture More Meaning", *ACM Trans. Database Syst. 4*, 4(Dec. 1979), 397-434.

[GMN84] Gallaire, H., Minker, J., and Nicolas, J. M. 1984. "Logic and Databases: A Deductive Approach," *Computing Surveys*, vol. 16, no. 2, June 1984. pp. 151-185.

[IL84] Imielinski, T. and Lipski, W. "Incomplete Information in Relational Databases", *JACM*, vol. 31, no. 4, Oct. 1984, pp. 761-791.

[Mi87] Minker, J. "Perspectives in Deductive Databases", invited talk in the 6th *ACM* PODS, San Diago, also Technical Report UMIACS-TR-87-7, University of Maryland, March 1987.

[Re86] Reiter, R. "A Sound and Sometimes Complete Query Evaluation Algorithm for Relational Databases with Null Values". *JACM*, vol. 33, no. 2, April 1986, pp. 349-370.

[Ul82] Ullman, J. D. *Principles of Database Systems*, Computer Science Press, Potomac, Maryland, 1982.

[Var85] Vardi, M. Y. "Querying Logical Databases". In *Proceeding of the 4th ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, (Portland, Ore., Mar. 25-27). *ACM*, New York 1985, pp. 57-65.

[Vas79] Vassiliou, Y. "Null Values in Data Base Management: A Denotational Semantics Approach", In *Proceedings of the ACM SIGMOD 1979 International Conference on Management of Data* (Boston, Mass., May 30-June 1). *ACM*, New York, 1979, pp. 162-169.

[YC87] Yuan, Y. L. and Chiang, D. A. "A Sound and Complete Query Evaluation Algorithm for Relational Databases with Null Values," The Center for Advanced Computer Studies, Technical Report, University of Southwestern Louisiana, July 1987.

[YH85] Yahya, A. and Henschen, L. "Deduction in Non-Horn Databases", *Journal of Automated Reasoning*, 1 (1985) pp. 141-160.