SOLVING IMPLICATION PROBLEMS IN DATABASE APPLICATIONS

Xian-He Sun, Nabil Kamel, and Lionel M. Ni
Department of Computer Science
Michigan State University
East Lansing, MI 48824-1027

Abstract: Computing queries from derived relations, optimizing queries from a group of queries, and updating materialized views are important database problems and have attracted much attention. One thing common to these problems is their demand to quickly solve the implication problem -- given two predicates σ_Q and σ_T, can σ_Q imply σ_T ($\sigma_Q \rightarrow \sigma_T$)? The implication problem has been solved by converting it into a satisfiability problem. Based on a graph representation, a detailed study of the general implication problem on its own is presented in this paper. We proved that the general implication problem, in which all six comparison operators: $=, \neq, <, >, \leq, \geq$, as well as conjunctions and disjunctions are allowed, is NP-hard. In the case when "$\neq$" operators are not allowed in σ_Q and disjunctions are not allowed in σ_T, a polynomial time algorithm is proposed to solve this restricted implication problem. The influence of the "$\neq$" operator and disjunctions are studied. Our theoretical results show that for some special cases the polynomial complexity algorithm can solve the implication problem which allows the "$\neq$" operator or disjunctions in the predicates. Necessary conditions for detecting when the "$\neq$" operator and disjunctions are allowed are also given. These results are very useful in creating heuristic methods.

I. INTRODUCTION

The problem of querying fragments of relations is of fundamental importance in many database applications. This problem is known as the *derivability problem* and can be formally defined as follows. Given a query Q and a set of d stored fragments $T_1, T_2, \cdots, T_d$, can the query Q be computed from these d fragments? The fragments could be, for example, a set of temporaries in a distributed database system or a set of prestored main memory query results. If Q and $T_1, T_2, \cdots, T_d$ are formed by selections and joins, then Q and $T_1, T_2, \cdots, T_d$ are reducible to Boolean expressions, i.e. *predicates*. In this case, the derivability problem becomes the *implication problem*.

In the database community, the derivability problem has gained much attention, especially in the area of distribute database systems, where the communication cost is a major concern. The derivability problem has been studied in the past [1], [2], [12], [13]. However, no efficient method has been obtained. As pointed out by Jarke and Koch [5], there seems to have no coherent theory in this area yet. It is known that for *Projection-Selection-Join* (PSJ) expressions (expressions in which the operations only involve project, select, join, and Cartesian product), the derivability entails processing implication [2].

Several other database applications also require the processing of predicates implication. The performance of global optimization protocols for global query optimization proposed by Finkelstein and Sellis [1], [11] is greatly dependent on the efficiency of processing predicates implication. The views in relational database which are physically stored are called *materialized views*. When a relational database is updated, it has to determine which materialized views have to be updated. The finding of these materialized views also requires implication processing [13],[14]. In a horizontally partitioned database system, as indicated in [9], [10], implications between predicates are relevant to the determination of accesses to candidate fragments. The new query optimization approaches using the page-query and page-node structures to retain the results of some queries as an aid in processing subsequent queries rely on processing implications on the prestored results [6],[7],[8]. Several problems in database design including access paths, consistency of integrity constraints, and distribution of data can all be treated as predicates implication problem [12].

Let σ_Q and σ_T be predicates with variables $x_1, x_2, \cdots, x_n$. The implication problem is traditionally solved in the form of *satisfiability problem* through the follow equivalence equation [1], [2], [12], [13].

$$\forall x_1, x_2, \cdots, x_n (\sigma_Q \rightarrow \sigma_T)$$
$$\Leftrightarrow \exists x_1, \cdots, x_n (\neg \sigma_T \wedge \sigma_Q) \qquad (1.1)$$

Here and throughout, we use $\vee$, $\wedge$ and $\neg$ to represent logic OR, AND, and NOT operations, respectively. Also, "$\Leftrightarrow$" and "$\equiv$" denote "if and only if" and "equivalence", respectively. If each of σ_Q and σ_T is a conjunction of one or two variable comparisons, σ_Q consists of n variables (or attributes in database term), and σ_T consists of k comparisons ($\sigma_T = B_1 \wedge B_2 \wedge \cdots \wedge B_k$), then (1.1) can be rewritten as:

$$\forall x_1, \cdots, x_n(\sigma_Q \rightarrow \sigma_T)$$
$$\Leftrightarrow \exists x_1, \cdots, x_n(\neg\sigma_T \wedge \sigma_Q)$$
$$\Leftrightarrow \exists x_1, \cdots, x_n(\neg B_1 \wedge \sigma_Q \vee \neg B_2 \wedge \sigma_Q \vee \cdots \vee \neg B_k \wedge \sigma_Q)$$
$$\Leftrightarrow \bigwedge_{i=1}^{k} \exists x_1, \cdots, x_n(\neg B_i \wedge \sigma_Q) \qquad (1.2)$$

From (1.2), the problem of solving one implication problem becomes the solving of k *satisfiability problems*. Rosenkrantz and Hunt [3] have shown that if the "$\neq$" operator is not allowed in ($\neg B_i \wedge \sigma_Q$), the time complexity for each satisfiability problem, ($\neg B_i \wedge \sigma_Q$, for $1 \le i \le k$), is $O(n^3)$. Therefore, the time complexity for the implication problem shown in (1.2) is $O(n^3 k)$.

This paper is organized as follows. In Section II, we introduce the basic concept and some definitions and terminologies of the implication problem. The implication problem is modeled as a weighted directed graph. In Section III, we prove that the general implication problem ($\sigma_Q \rightarrow \sigma_T$?) is NP-hard. Consequently, we prove that the derivability problem is NP-hard. Section IV presents an $O(n^3 + k)$ time complexity algorithm for the restricted implication problem in which the predicates consist of a conjunction of one or two attribute comparisons and σ_Q does not contain the "$\neq$" operator. Necessary conditions for solving the general implication problem when "$\neq$" comparisons are allowed in σ_Q or disjunctions are allowed in σ_T are developed in Section V. Using these conditions, we show that in some special cases the polynomial complexity algorithm can solve the general implication problem. Results presented in Section V are useful in creating heuristic methods. Due to page limitation, detailed proof of theorems is ignored in this short paper and will be presented in the full paper.

II. DEFINITIONS AND TERMINOLOGIES

Based on Rosenkrantz and Hunt in [3], a weighted directed graph structure is used to model the implication problem. This section gives definitions and terminologies used in this paper. Some terminologies defined in [3] are repeated here.

Definition 1: A logic expression σ_Q implies another logic expression σ_T ($\sigma_Q \rightarrow \sigma_T$) if and only if every assignment that maps σ_Q into the true statement also maps σ_T into the true statement.

If the logic expressions σ_Q and σ_T represent the selection condition of a general PSJ query, then the above definition means that $\sigma_Q \rightarrow \sigma_T$ if and only if every tuple obtained by evaluating σ_Q can be obtained by evaluating σ_T.

Definition 2: A logic expression σ_Q is satisfiable if there exists an assignment that maps σ_Q into the true statement. The problem of checking if a logic expression is satisfiable is called the *satisfiability problem*.

In database all attributes belong to some countable domains. Thus, all attributes in a database can be considered to have a subset of integers as their domain. In the following discussions we assume that all attributes have the set of all integers as their domains, although, as will be shown later, this assumption of unbounded domains can be removed with no effect on the computational complexity.

A comparison operator is one of the following six operators: "=", "$\neq$", "<", "$\le$", ">", and "$\ge$".

Definition 3: A comparison is a *simple comparison* if it involves no more than two attributes and is in one of the following comparison forms:

1. A single-attribute simple comparison

< attribute > < comparison operator > < constant expression >
< constant expression > < comparison operator > < attribute >

2. A double-attributed simple comparison

< attribute > < comparison operator > < attribute >
+ < constant expression >
< attribute > + < constant expression >
< comparison operator > < attribute >

In Definition 3, the constant expression is an arithmetic expression consisting of constants, where a constant can be a zero, a negative integer, or a positive integer. In the following discussions, in addition to those numerical values we shall use a, b, c, d, and e to represent constants and all other variables are attributes. The following five examples are simple comparisons:

$y \le -3$
$x > y + c$
$w \neq v$
$27 \ge x$
$x < y + (c-1)$.

Definition 4: A logic expression is called a *conjunctive mixed predicate* if it is a conjunction of simple comparisons.

Definition 5: A predicate with no "$\neq$" operator involved is referred to as an *unequal-free predicate*.

Definition 6: A conjunctive mixed predicate is said to be *normalized* if it contains only "$\le$" comparisons. A conjunctive mixed predicate is said to be *seminormalized* if it contains only "$\le$" and "$\neq$" comparisons.

The following lemma is useful in establishing the relationship between these predicates defined above.

Lemma 1: Any conjunctive unequal-free mixed predicate can be converted, in linear time, into an equivalent normalized predicate, and any conjunctive mixed predicate can be converted, in linear time, into an equivalent seminormalized predicate.

Proof: A conjunctive unequal-free mixed predicate may involve five different comparison operators: "=", "<", "≤", ">", and "≥". We have to show that these operators can be rewritten using the "≤" operator and conjunctions. For the case of double-attribute simple expression, the following equivalence relations are always true.

$$[x < y + c] \equiv [x \le y + (c-1)]$$
$$[x > y + c] \equiv [y \le x + (-c-1)]$$
$$[x = y + c] \equiv [(x \le y + c) \wedge (y \le x + (-c))]$$
$$[x \ge y + c] \equiv [y \le x + (-c)]$$

The equivalence relations can be similarly derived for the case of single-attribute simple expression. By induction on the number of simple comparisons, any conjunctive unequal-free mixed predicate can be converted into a semi-normalized predicate in linear time. Since the "≠" operator cannot be expressed in terms of "≤" and conjunctions, any conjunctive mixed predicate can be converted into a semi-normalized predicate in linear time. ■

Because a normalized predicate consists of the "≤" operator only, it can be represented as a weighted digraph, where nodes represent attributes and directed edges represent "≤" comparisons. In other words, each node in the digraph represents an attribute in the predicate, and a directed edge from node x to node y with weight c represents the predicate $x \le y+c$. A special node labelled "0" is used for single-attribute predicates. Note that a node may have multiple edges if the corresponding attribute is involved in many predicates. The edge constructed by the comparison $x \le y+c$ is referred to as edge $x \le y+c$. The weight of a path (or a cycle) in a directed graph is the accumulated sum of the weights of those edges constituting the path (or cycle).

Definition 7: An implication problem $(\sigma_Q \rightarrow \sigma_T?)$ is called a *restricted implication problem* if σ_Q is a conjunctive unequal-free mixed predicate and σ_T is a conjunctive mixed predicate.

From Definition 6, a restricted implication problem $(\sigma_Q \rightarrow \sigma_T?)$ has a normalized predicate σ_Q and a seminormalized predicate σ_T. The next section will prove that the general implication problem is NP-hard. Then in Section IV, we propose an efficient algorithm for solving the restricted implication problem.

III. THE GENERAL IMPLICATION PROBLEM

A general implication problem $(\sigma_Q \rightarrow \sigma_T?)$ have two predicates σ_Q and σ_T which may involve six comparison operators: "=", "≠", "<", "≤", ">", and "≥" as well as conjunctions and disjunctions. We shall prove that the general implication problem is NP-hard. This leads to a conclusion that the more general problem, the derivability problem, is NP-hard.

Theorem 1: If disjunctions are allowed in σ_T, the implication problem $(\sigma_Q \rightarrow \sigma_T?)$ is NP-hard.

Proof: The following Boolean satisfiability problem [15] is known to be NP-hard. "Given m clauses $C_1, C_2, \cdots, C_m$, where $C_i = (x_{i_1} \vee x_{i_2} \vee \cdots \vee x_{i_k})$ for $1 \le i \le m$ involving n Boolean variables $x_1, x_2, \cdots, x_n$, the question of whether the formula $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ is satisfiable is NP-complete". We prove this theorem by showing that the Boolean satisfiability problem can be reduced polynomially to the implication problem with disjunctions allowed in σ_T.

Given a Boolean formula $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ with Boolean variables $x_1, x_2, \cdots, x_n$, the following two steps are used to reduce the formula.

(1) Choose a Boolean variable x_{n+1} such that $x_{n+1} \notin \{x_1, x_2, \cdots, x_n\}$. Thus, $C_1 \wedge C_2 \wedge \cdots \wedge C_m$ is satisfiable if and only if $C_1 \wedge C_2 \wedge \cdots \wedge C_m \wedge x_{n+1}$ is satisfiable.

(2) Replace each Boolean variable x_i by a simple comparison $u_i \le v_i$. $\bar{x}_i$, the complement of x_i, is replaced by $\neg (u_i \le v_i)$.

We denote the clause C_i, with the Boolean variables replaced, by R_i. Since there is no dependence among the variables, and by choosing different values of u_i and v_i, we can get each simple comparison $u_i \le v_i$ to be true or false. Thus, we have

$$C_1 \wedge C_2 \wedge \cdots \wedge C_m$$

is satisfiable if and only if

$$R_1 \wedge R_2 \wedge \cdots \wedge R_m \wedge (u_{i+1} \le v_{i+1}) \qquad (3.1)$$

is satisfiable. (3.1) is satisfiable if and only if

$$(u_{i+1} \le v_{i+1}) \rightarrow (\neg R_1 \vee \neg R_2 \vee \cdots \vee \neg R_m)$$

is not true [10]. Note that each $\neg R_i$ is a conjunctive mixed predicate. The satisfiability of each $\neg R_i$ can be checked in polynomial time [3]. If for some i, $\neg R_i$ is unsatisfiable, then the $\neg R_i$ is removed from the implication problem. If all R_i's (for $i=1,2, \cdots, m$) have been removed, then the implication is not true and we are done. In this way, the Boolean variable satisfiability problem is reduced to the implication problem. Thus, if disjunctions are allowed in σ_T, the implication problem is NP-hard. ■

Theorem 2: If the "≠" operator is allowed in σ_Q, the implication problem $(\sigma_Q \rightarrow \sigma_T?)$ is NP-hard.

Proof: We show that the satisfiability problem of conjunctive mixed predicates with "≠" operators allowed can be solved by the implication problem $(\sigma_Q \rightarrow \sigma_T?)$ with "≠" operators allowed in σ_Q. By Rosenkrantz and Hunt [3] the former is NP-hard, so the latter is also NP-hard.

Suppose that we have a seminormalized predicate $\sigma = A_1 \wedge A_2 \wedge \cdots \wedge A_n$, where A_i's (for $i=1,2, \cdots, n$) are simple comparisons with the "≠" operator. The satisfiability of σ can be checked by the following steps.

(1) If one of the A_i's is unsatisfiable, then σ is unsatisfiable.

(2) Assume that we are given a seminormalized selection A and a simple comparison A_i. If both A and A_i are satisfiable, then $A \rightarrow A_i$ is not true if and only if $\neg A_i \wedge A$ is satisfiable. The satisfiability problem can be solved

by checking the implication problem based on the following procedure.

$A := A_1$
for $i = 2$ to n **do**
  if $A \rightarrow \neg A_i$ is not true **then** $A := A \wedge A_i$
  else return(σ is unsatisfiable);
return(σ is satisfiable);

Step (1) can be solved in linear time. Step (2) is actually the solving of the implication problem $n-1$ times. Since the satisfiability problem involving the "$\neq$" operator is known to be NP-hard [3], the implication problem is NP-hard. ■

Theorem 1 and Theorem 2 show that the general implication problem is NP-hard. As the general implication problem is a special case of the derivability problem, by the lower bound property we have the following result.

Corollary 1: The derivability problem is NP-hard.

In the next section, we shall show that if "$\neq$" comparisons are not allowed in σ_Q and disjunctions are not allowed in σ_T, this restricted implication problem can be efficiently solved.

IV. THE RESTRICTED IMPLICATION PROBLEM

This section considers the restricted implication problem ($\sigma_Q \rightarrow \sigma_T$?) in which σ_Q is a conjunctive unequal-free mixed predicate and σ_T is a conjunctive mixed predicate. Note that according to Definition 6, σ_Q and σ_T are expressed in normalized predicate and seminormalized predicate, respectively. Let $G(\sigma_Q)$ be the corresponding weighted digraphs of σ_Q. We assume that no multiple edges exist between two nodes in $G(\sigma_Q)$. If there were multiple edges, the graph constructing algorithm will only keep the edge with the smallest weight and remove all other edges. Furthermore, both σ_Q and σ_T are assumed satisfiable. Determination of the satisfiability of a normalized predicate is stated in Theorem 3 which is due to Rosenkrantz and Hunt [3].

Theorem 3: For any normalized predicate σ_Q, σ_Q is unsatisfiable if and only if there is a negative cycle in $G(\sigma_Q)$.

Lemma 2: For any normalized predicate σ_Q, $\sigma_Q \rightarrow u \leq v + c$ if and only if there is a path in $G(\sigma_Q)$ from node u to node v with weight less than or equal to c.

Proof: Since σ_Q and $u \leq v+c$ are satisfiable, by mathematical logic [10], $\sigma_Q \rightarrow (u \leq v + c)$ if and only if

$$\neg(u \leq v + c) \wedge \sigma_Q = \varnothing \tag{4.1}$$

Here $\varnothing$ represents an empty set or the logical expression is unsatisfiable. By Theorem 3, $\neg(u \leq v + c) \wedge \sigma_Q$ is unsatisfiable if and only if there exists a negative cycle in the graph of $(v \leq u - c - 1) \wedge \sigma_Q$. Note that $(v \leq u - c - 1) \equiv \neg(u \leq v + c)$. Since σ_Q is satisfiable, there is no negative cycle in $G(\sigma_Q)$. Thus, the negative cycle must contain the edge $(v \leq u - c - 1)$. This implies the existence of a path in $G(\sigma_Q)$ from node u to node v with weight less than or equal to c. ■

Lemma 3: For any normalized predicate σ_Q and any simple comparison $u \neq v+c$, $\sigma_Q \rightarrow (u \neq v + c)$ if and only if one of the following conditions occurs:

(1) There is a path in $G(\sigma_Q)$ from node u to node v with weight less than or equal to $c-1$,
(2) There is a path in $G(\sigma_Q)$ from node v to node u with weight less than or equal to $(-c-1)$.

Proof: By mathematic logic [10],

$$\begin{aligned} & \sigma_Q \rightarrow u \neq v + c \\ \Leftrightarrow\ & \sigma_Q \rightarrow (u \leq v + c - 1) \vee (v \leq u - c - 1) \\ \Leftrightarrow\ & \neg(u \leq v + c - 1) \wedge \sigma_Q \rightarrow (v \leq u - c - 1) \end{aligned} \tag{4.2}$$

By (4.2) and Lemma 2, the "if" part is trivial. Here we examine the "only if" part. If $\neg(u \leq v+c-1) \wedge \sigma_Q = \varnothing$, then $\sigma_Q \rightarrow u \leq v+c$. By Lemma 2, the condition (1) is proven.

If $(v \leq u-c) \wedge \sigma_Q$, where $(v \leq u-c) \equiv \neg(u \leq v+c-1)$, is satisfiable, then, by Lemma 2, there is a path from node v to node u with weight less than or equal to $(-c-1)$ in $G((v \leq u-c) \wedge \sigma_Q)$. This path cannot contain the edge $v \leq u-c$; otherwise, there is a negative cycle in $G(\sigma_Q)$ which contradicts with the fact that σ_Q is satisfiable. Thus, the condition (2) is proven. ■

Lemma 4: For any normalized predicate σ_Q and any seminormalized predicate σ_T, $\sigma_Q \rightarrow \sigma_T$ if and only if for any simple comparison $u \leq v+c$ (or $u \neq v+c$) in σ_T, $u \leq v+c$ (or $u \neq v+c$) is implied by σ_Q.

Proof: Let σ_Q, σ_T consist of n variables and $\sigma_T = B_1 \wedge B_2 \wedge \cdots \wedge B_k$, where B_i's are simple comparisons, the following equivalence holds:

$$\begin{aligned} & \forall x_1, \cdots, x_n (\sigma_Q \rightarrow \sigma_T) \\ \Leftrightarrow\ & \forall x_1, \cdots, x_n (\sigma_Q \rightarrow B_1 \wedge B_2 \wedge \cdots \wedge B_k) \\ \Leftrightarrow\ & \forall x_1, \cdots, x_n [(\sigma_Q \rightarrow B_1) \wedge (\sigma_Q \rightarrow B_2) \wedge \cdots \wedge (\sigma_Q \rightarrow B_k)] \\ \Leftrightarrow\ & [\forall x_1, \cdots, x_n (\sigma_Q \rightarrow B_1)] \wedge [\forall x_1, \cdots, x_n (\sigma_Q \rightarrow B_2)] \wedge \cdots \\ & \wedge [\forall x_1, .., x_n (\sigma_Q \rightarrow B_k)] \end{aligned} \tag{4.3}$$

This is true if σ_T is a conjunctive mixed predicate and σ_Q is in any form. If σ_Q is a normalized predicate and σ_T is a seminormalized predicate, this lemma is proven by the above equivalence. ■

Theorem 4 is a direct result of Lemma 2, Lemma 3, and Lemma 4.

Theorem 4: For any normalized predicate σ_Q and any seminormalized predicate σ_T, $\sigma_Q \rightarrow \sigma_T$ if and only if for any simple comparison $u \leq v+c$ in σ_T, there is a path in $G(\sigma_Q)$ from node u to node v with weight less than or equal to c, and for any simple comparison $u \neq v+c$ in σ_T, one of the following conditions occurs:

(1) There is a path in $G(\sigma_Q)$ from node u to node v with weight less than or equal to $c-1$,

(2) There is a path in $G(\sigma_Q)$ from node v to node u with weight less than or equal to $(-c-1)$.

Now we are ready to present an algorithm to the solution of the restricted implication problem. The input to this algorithm, RESTRICTED-IMPLICATION-CHECK, is a seminormalized predicate σ_T and the graph of a normalized predicate σ_Q. The output of this algorithm is YES if $\sigma_Q \rightarrow \sigma_T$; otherwise, the output is NO.

Algorithm RESTRICTED-IMPLICATION-CHECK

S1: If an attribute u appears in σ_T, but u is not a node in $G(\sigma_Q)$, then **return**(NO).

S2: Using Floyd's Algorithm to find the weight of the shortest path for each pair of nodes in $G(\sigma_Q)$.
If a negative cycle is found in running Floyd's algorithm, then **return**(YES).

S3: For each comparison $u \le v+c$ in σ_T, check the result obtained in S2.
If the shortest path from node u to node v in $G(\sigma_Q)$ has its path weight greater than c, then **return**(NO).

S4: For each comparison $u \ne v+c$ in σ_T
If there is a shortest path from node u to node v in $G(\sigma_Q)$ with its path weight $> c-1$ and there is a shortest path from node v to node u in $G(\sigma_Q)$ with its path weight $> (-c-1)$,
then **return**(NO).

S5: **return**(YES).

We denote the number of comparisons of σ_i by E_i and denote the number of attributes in σ_i by V_i for $i=T,Q$. Here E_Q is greater than or equal to the number of edges in $G(\sigma_Q)$ and V_Q+1 is equal to the number of nodes (vertices) in $G(\sigma_Q)$ including the node labelled "0".

S1 is a quick scan to eliminate some cases where the implication is trivially unsatisfied. Its complexity is $O(V_T V_Q)$. Floyd's algorithm which can be found in most graph algorithm books ([4], for example) is used to find the weight of the shortest path for each pair of nodes. The complexity of S2 is $O(V_Q^3)$. If a negative cycle is found, then σ_Q is unsatisfiable (Theorem 3) and the implication is true. The complexity of S3 is $O(E_T)$. The complexity of S4 is also $O(E_T)$. Thus, the complexity of the algorithm is $O(E_T+V_Q^3)$. For $V_Q=n$ and $E_T=k$ the complexity of the algorithm is $O(n^3+k)$ which is significantly less than $O(n^3k)$ if the problem is solved based on the satisfiability problem. If we assume that $E_T \le V_Q^3$, which is almost always true in real cases, then the complexity of the algorithm is $O(n^3)$. Based on Lemma 1, Theorem 4, and the above discussions, we have the following main result.

Theorem 5: The restricted implication problem ($\sigma_Q \rightarrow \sigma_T$?) can be solved in $O(n^3+k)$, where n is the number of attributes in σ_Q and k is the number of comparisons in σ_T.

In order to simplify the proof, we have assumed that the domain of each attribute is the set of all integers. In fact, this assumption is unnecessary. The algorithm presented above can be used for attributes having a set of consecutive integers, either bounded or unbounded, as their domain. The only thing we need to do is to modify the predicates σ_T and σ_Q by adding the domain constraints into the graph $G(\sigma_Q)$. Let x_i be the attribute appearing in σ_T with domain $[a_i,b_i]$ for $1 \le i \le n$. If some of the x_i's do not appear in $G(\sigma_Q)$, then the implication does not hold. Otherwise, we begin adding edges. For each x_i, by the restriction $a_i \le x_i \le b_i$ in the following ways. If b_i is finite, we add an edge from node x_i to node 0 with weight b_i into $G(\sigma_T)$. If a_i is finite, we add an edge from node 0 to node x_i with weight $-a_i$ into $G(\sigma_T)$. Similarly, we add edges into $G(\sigma_Q)$ for each attribute appearing in σ_Q. Since the algorithm presented above can solve the restricted implication problem on predicates with unbounded domains based on the modified graph , the restricted implication problem with attributes having bounded domains is readily solved.

Now let's consider some applications of the proposed RESTRICTED-IMPLICATION-CHECK algorithm. Consider the following frequently asked database question. Given a group of fragment predicates $\sigma_{T_1}, \sigma_{T_2}, \cdots, \sigma_{T_d}$ and one query predicate σ_Q, does there exist an i $(1 \le i \le d)$ such that $\sigma_Q \rightarrow \sigma_{T_i}$ is true? In the above algorithm, we only run Floyd's algorithm to find the shortest path of each pair of nodes in $G(\sigma_Q)$. For each comparison in σ_T we simply compare their offset with the weight of the shortest path in $G(\sigma_Q)$. This property makes the algorithm much more efficient for the solution of the above question. If predicate σ_{T_i} has k_i comparisons for $i=1,\cdots,d$, then instead of $O((\sum_{i=1}^{d} k_i)n^3)$ time complexity obtained by solving the problem by satisfiability, the algorithm we proposed reduces the time complexity to $O(n^3+\sum_{i=1}^{d} k_i)$.

In the case when disjunctions are used in σ_Q ($\sigma_Q \equiv \sigma_{Q_1} \vee \cdots \vee \sigma_{Q_d}$, where σ_{Q_j}'s are conjunctive unequal-free mixed predicates), the proposed algorithm can still be applied. Disjunctions in σ_Q can be handled by showing that the implication is true if any of the σ_{Q_j}'s (for $1 \le j \le d$) in σ_Q is true. If each σ_{Q_j} has n attributes, then the time complexity is $O(d(n^3+k))$, where k is the number of comparisons in σ_T. However, if conjunctions are used instead of disjunctions, i.e., $\sigma_Q \equiv \sigma_{Q_1} \wedge \cdots \wedge \sigma_{Q_d}$, the corresponding time complexity will be increased to $O((dn)^3+k)$.

Another important feature of the proposed algorithm is the ability to allow the "$\ne$" operator in σ_T. It is known that the "$\ne$" operator makes the satisfiabilty problem very difficult to solve [3] should it occur in the predicates. However, as indicated in Lemma 3, our proposed algorithm does allow the occurrence of the "$\ne$" operator in σ_T.

In the next section, we shall show that in some special cases, our proposed algorithm can solve the general impli-

cation problem in which "≠" operators are used in σ_Q or disjunctions are used in σ_T.

V. THE INFLUENCE OF ≠ OPERATORS AND DISJUNCTIONS

In this section we show that the "≠" operator in σ_Q sometime have no influence on the implication problem. In this case, the "≠" comparison can be ignored. The influence of disjunctions in σ_T is also studied.

Theorem 6: Let σ be a normalized predicate. If $(x \neq y + c')\wedge\sigma$ is satisfiable and $((x \neq y + c')\wedge\sigma)\rightarrow(u \leq v + c)$, then $\sigma\rightarrow(u \leq v + c + 1)$.

Proof: Since

$$(x \neq y + c') \equiv ((x \leq y + c' - 1)\vee(y \leq x - c' - 1))$$

we have

$$((x \neq y + c')\wedge\sigma) \equiv ((x \leq y + c' - 1\wedge\sigma)\vee(y \leq x - c' - 1\wedge\sigma))$$

Consider the following three cases.

(1) If $(x \leq y + c' - 1)\wedge\sigma$ is unsatisfiable, then

$$\sigma\rightarrow\neg(x \leq y + c' - 1),$$

where $\neg(x \leq y + c' - 1) \equiv y \leq x - c'$. Since $(x \neq y + c')\wedge\sigma$ is satisfiable, we must have $(y \leq x - c' - 1)\wedge\sigma$ satisfiable and

$$((y \leq x - c' - 1)\wedge\sigma)\rightarrow(u \leq v+c).$$

Thus,

$$((y \leq x - c')\wedge\sigma)\rightarrow(u \leq v + c + 1).$$

Because $\sigma\rightarrow(y \leq x - c')$, it follows that $\sigma \equiv (y \leq x - c')\wedge\sigma$ and $\sigma\rightarrow(u \leq v + c + 1)$.

(2) If $(y \leq x - c' - 1)\wedge\sigma$ is unsatisfiable, then

$$\sigma\rightarrow\neg(y \leq x - c' - 1),$$

where $\neg(y \leq x - c' - 1) \equiv (x \leq y + c')$. By a similar argument as (1), we have

$$\sigma\rightarrow(u \leq v + c + 1).$$

(3) If both $(y \leq x - c' - 1)\wedge\sigma$ and $(x \leq y + c' - 1)\wedge\sigma$ are satisfiable, then

$$((y \leq x - c' - 1)\wedge\sigma)\rightarrow(u \leq v + c)$$

and

$$((x \leq y + c' - 1)\wedge\sigma)\rightarrow(u \leq v + c)$$

By Lemma 2 there exists a path p_1 from u to v with weight less than or equal to c in the weighted digraph $G((y \leq x - c' - 1)\wedge\sigma)$. If p_1 does not contain the edge $(y \leq x - c' - 1)$, by Lemma 3 we have $\sigma\rightarrow(u \leq v + c)$ and the theorem is proved. Now we assume that p_1 contains the edge $(y \leq x - c' - 1)$. Furthermore, we assume that in the path p_1, the weight from node u to node y is b and the weight from node x to node v is e. By the same reason there exists a path p_2 from u to v with weight less than or equal to c in $G((x \leq y + c' - 1)\wedge\sigma)$. Similarly, we assume that p_2 contains the edge $(x \leq y + c'-1)$. Furthermore, we assume that in the path p_2, the weight from node u to node x is a and the weight from node y to node v is d. The relation of these nodes and paths is shown in Figure 1.

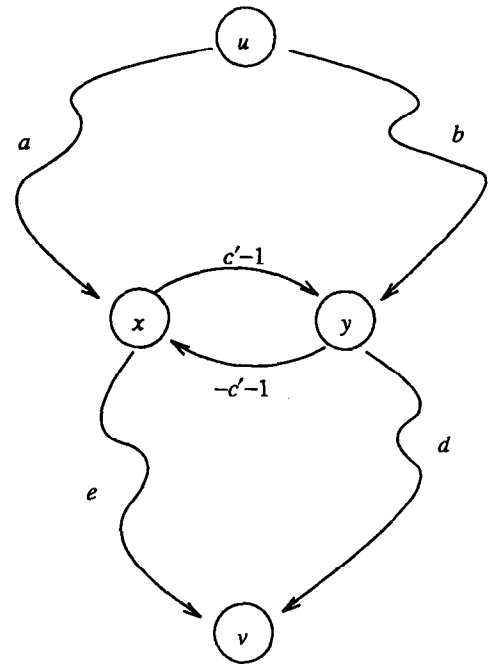

Figure 1. The graph representation of the paths in Theorem 6.

If $\sigma\rightarrow(u \leq v + c)$ is true, then $\sigma\rightarrow(u \leq v + c + 1)$ is true and we are done. If $\sigma\rightarrow(u \leq v + c)$ is not true, we have the following inequalities:

$$b+d>a+c'-1+d$$

$$a+e>b-c'-1+e$$

$$b+d>b-c'-1+e$$

$$a+e>a+c'-1+d$$

Subsequently, we have

$$a-b\leq -c'$$

$$a-b\geq -c'$$

$$e-d\leq c'$$

$$e-d\geq c'$$

Thus, we obtain $b-a=c'$ and $e-d=c'$ which result in $a+e=a+c'+d$. Since $a+c'-1+d \leq c$, we have $a+e \leq c+1$. Thus there is a path from u to v with weight less than or equal to $c+1$ in $G(\sigma)$. By Lemma 2 we have $\sigma\rightarrow(u \leq v + c + 1)$. ■

Theorem 6 can be generalized to seminormalized predicates containing more than one "≠" comparisons.

Corollary 2: If $\sigma_Q \equiv (\sigma_1 \wedge \sigma_2)$ is satisfiable and $\sigma \rightarrow (u \le v + c)$, where σ_1 is a conjunction of m "$\neq$" simple comparisons and σ_2 is a normalized predicate, then $\sigma_2 \rightarrow (u \le v + c + m)$.

Proof: This proof is done by induction on the number of "$\neq$" comparisons in σ_1 and using the result of Theorem 6. ■

From Lemma 2, Lemma 3, and Corollary 2, we have the following result.

Corollary 3: If $\sigma_Q \equiv (\sigma_1 \wedge \sigma_2)$ is satisfiable and $\sigma_Q \rightarrow \sigma_T$, where σ_1 is a conjunction of m "$\neq$" simple comparisons, σ_2 is a normalized predicate, and σ_T is a seminormalized predicate, the following hold:

(1) For any simple comparison $u \le v+c$ in σ_T, $\sigma_2 \rightarrow (u \le v+c+m)$,

(2) For any simple comparison $x \neq y+c'$ in σ_T, $\sigma_2 \rightarrow (x \le y+c'+m-1)$ or $\sigma_2 \rightarrow (y \le x-c'+m-1)$.

Corollary 3 may be used to solved a general implication problem ($\sigma_Q \rightarrow \sigma_T$?) when the "$\neq$" operator is involved in the predicate σ_Q. We can first process ($\sigma_2 \rightarrow \sigma_T$?). For each simple comparison $u \le v+c$ in σ_T, if $\sigma_2 \rightarrow (u \le v+c)$ then $u \le v+c$ can be removed from σ_T. If σ_2 does not imply $u \le v + c + m$, then the implication ($\sigma_Q \rightarrow \sigma_T$?) is not true. For each simple comparison $x \neq y+c'$ in σ_T, we have a similar statement for $x \le y+c'-1$ or $y \le x-c'-1$. In this way if the above conditions are satisfied, without processing the "$\neq$" comparisons in σ_Q, the implication problem ($\sigma_Q \rightarrow \sigma_T$?) can be solved. Based on this result, heuristic methods can be created.

Now we turn to the case when disjunctions are allowed in σ_T in the general implication problem ($\sigma_Q \rightarrow \sigma_T$?). Here we assume that the σ_Q is a satisfiable normalized predicate, σ_{T_1} and σ_{T_2} are satisfiable seminormalized predicates, and $\sigma_T \equiv (\sigma_{T_1} \vee \sigma_{T_2})$.

Theorem 7: If $\sigma_{T_1} \rightarrow \neg(u \le v + c+1)$ and $\sigma_Q \rightarrow (\sigma_{T_1} \vee (\sigma_{T_2} \wedge u \le v + c))$, then $\sigma_Q \rightarrow (u \le v + c)$ or $\sigma_Q \rightarrow \sigma_{T_1}$.

Proof: Since $\sigma_Q \rightarrow (\sigma_{T_1} \vee (\sigma_{T_2} \wedge u \le v + c))$, by mathematical logic [10], we have

$$(\neg(u \le v + c) \wedge \sigma_Q) \rightarrow \sigma_{T_1}.$$

If $\neg(u \le v + c) \wedge \sigma_Q$ is unsatisfiable, then

$$\sigma_Q \rightarrow (u \le v + c)$$

and we are done. If $\neg(u \le v + c) \wedge \sigma_Q$ is satisfiable, since

$$\sigma_{T_1} \rightarrow \neg(u \le v + c + 1),$$

where $\neg(u \le v + c + 1) \equiv (v \le u - c - 2)$ and

$$(\neg(u \le v + c) \wedge \sigma_Q) \rightarrow \sigma_{T_1},$$

we have

$$(\neg(u \le v + c) \wedge \sigma_Q) \rightarrow (v \le u - c - 2).$$

There is a path p from v to u with weight less than or equal to $(-c-2)$ in the weighted graph $G(\neg(u \le v + c) \wedge \sigma_Q)$. If the path p contains the edge $\neg(u \le v + c) \equiv (v \le u - c - 1)$, then σ_Q has a negative cycle (see Figure 2). This contradicts the assumption that σ_Q is satisfiable. If the path p does not contain the edge $v \le u - c - 1$, then

$$\sigma_Q \rightarrow (v \le u - c - 2).$$

Thus we have $(\neg(u \le v + c) \wedge \sigma_Q) \equiv \sigma_Q$, and $\sigma_Q \rightarrow \sigma_{T_1}$ is proved. ■

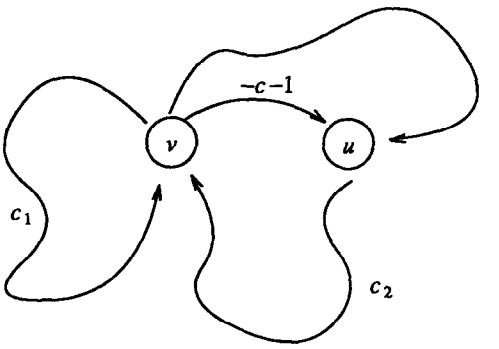

Figure 2. The graph representation of the paths in Theorem 7.

Since $\sigma_{T_1} \rightarrow \neg(u \le v + c + 1)$ is equivalent to having a negative cycle with weight less than -1 (Lemma 2) in $G(\sigma_{T_1} \wedge (u \le v + c))$. Theorem 7 can be restated in the following form.

Theorem 7': If $G(\sigma_{T_1} \wedge (u \le v + c))$ has a cycle with weight less than -1 and $\sigma_Q \rightarrow (\sigma_{T_1} \vee (\sigma_{T_2} \wedge (u \le v + c)))$, then $\sigma_Q \rightarrow \sigma_{T_1}$ or $\sigma_Q \rightarrow (u \le v + c)$.

Theorem 7 shows that to solve the implication problem $\sigma_Q \rightarrow (\sigma_{T_1} \vee \sigma_{T_2})$, we can do preprocessing on σ_{T_1} and σ_{T_2}. We first find all the simple comparisons $u \le v+c$ in σ_{T_1} such that $\sigma_{T_2} \rightarrow \neg(u \le v+c+1)$ is true. Then we find all the simple comparisons $u' \le v'+c'$ in σ_{T_2} such that $\sigma_{T_1} \rightarrow \neg(u' \le v'+c'+1)$ is true. If $\sigma_Q \rightarrow \sigma_{T_2}$ is true, we are done and the implication $\sigma_Q \rightarrow (\sigma_{T_1} \vee \sigma_{T_2})$ is true. If $\sigma_Q \rightarrow \sigma_{T_2}$ is not true, then $\sigma_Q \rightarrow (u \le v + c)$ must be true for all the comparisons found above. Otherwise, the implication $\sigma_Q \rightarrow (\sigma_{T_1} \vee \sigma_{T_2})$ is not true. Similar arguments can be stated for σ_{T_1} and all the comparisons $u' \le v'+c'$ found above. Thus, in these conditions, the implication problem $\sigma_Q \rightarrow (\sigma_{T_1} \vee \sigma_{T_2})$ can be solved without processing the disjunctions in σ_T. The following corollary is a direct result of Theorem 7'.

Corollary 4: $\sigma_Q \rightarrow (u \neq v+c)$ if and only if $\sigma_Q \rightarrow (u \le v + c - 1)$ or $\sigma_Q \rightarrow (v \le u - c - 1)$.

Proof: Since $(u \neq v + c) \equiv ((u \le v + c - 1) \vee (v \le u-c-1))$, the "if" part is trivial.

Since $(u \le v + c - 1) \wedge (v \le u - c - 1)$ forms a cycle with weight -2, the "only if" part follows from Theorem 7'. ■

Note that by Lemma 2, Corollary 4 is equivalent to Lemma 3. In other words, Corollary 4 is another proof for Lemma 3.

VI. CONCLUSION

This paper has addressed the problem of solving the implication problem ($\sigma_Q \rightarrow \sigma_T$?) which is frequently used in many database applications. A general implication problem involves "=", "≠", "<", "≤", ">", "≥" comparison operators, and conjunctions and disjunctions among predicates. We have proven that the general implication problem is NP-hard. For the restricted implication problem in which the "≠" operator is not allowed in σ_Q and disjunction is not allowed in σ_T, we proposed an $O(k+n^3)$, where k is the number of comparisons in σ_T and n is the number of attributes in σ_Q, algorithm to solve the problem. The proposed algorithm is much efficient than the traditionally used algorithm which has complexity $O(n^3k)$.

For the general implication problem, we have shown that in some conditions which the "≠" operator is allowed in σ_Q or disjunctions are allowed in σ_T, the problem can still be solved using the algorithm proposed in Section IV. These necessary conditions stated in Section V are useful in creating heuristic methods for solving general implication problems.

REFERENCES

[1] S. Finkelstein, "Common Expression Analysis in Database Applications", *Proceedings of the ACM SIGMOD Conference*, pp. 235-245, 1982.

[2] P.-Å. Larson and H.Z. Yang, "Computing Queries from Derived Relations", *Proc. of the 11th International Conference on Very Large Databases*, pp. 259-269, 1985.

[3] D.J. Rosenkrantz and H.B. Hunt III, "Processing Conjunctive Predicates and Queries," *Proc. of the 6th International Conference on Very Large Databases*, pp. 64-72, 1980.

[4] S. Even, *Graph Algorithms*, Computer Science Press, 1979

[5] M. Jarke and J. Koch, "Query Optimization in Database Systems", *ACM Computing Surveys*, Vol. 16, No. 2, pp. 111-152, June 1984.

[6] M. Jarke, "Common Subexpression Isolation in Multiple Query Optimization", *Query Processing in Database Systems*, W. Kim, D. Reiner, and D. Batory, Eds., Springer, New York.

[7] N.N. Kamel, "The Use of Controlled Redundancy in Self-Adaptive Databases", *Ph.D. Thesis*, Computer Science Department, The University of Colorado, Boulder, Colorado, (August 1985).

[8] N.N. Kamel, "Performance Enhancements of Rete Networks Through the Use of Page-Nodes", *Proc. of the 3rd IEEE Conference on Artificial Intelligence Applications*, Orlando, Fla., 1987.

[9] N.N. Kamel and R. King, "Optimal Management of Temporaries in Distributed Databases", *Proc. of the Second International Conference on Supercomputing*, 1986.

[10] H. Hermes, *Introduction to Mathematic Logic*, Springer - Verlag Pub. Co.

[11] T.K. Sellis, "Global Query Optimization", *ACM processings of SIGMOD'86*, pp. 191-205, 1986.

[12] R. Munz, H.-J. Schneider and F.Steyer, "Application Of Sub-Predicate Tests In Database Systems", *Proc. Fifth Int. Conf. on Very Large Data Bases*, pp. 426-435, Oct. 1979.

[13] J.A. Blakeley, N. Coburn and P.-Å. Larson, "Updating Derived Relations: Detecting Irrelevant and Autonomously Computable Updates", *Proc. Fifth Int. Conf. on Very Large Data Bases*, Kyoto, pp. 457-466, 1986.

[14] J.A. Blakeley, P.A. Larson and F.W. Tompa, "Efficiently Updating Materialized Views", *ACM Proceedings of SIGMOD '86*, pp. 61-71, 1986.

[15] C.H. Papadimitriou and K. Steiglitz, *Combinational Optimization Algorithm And Complexity*, Prentice-Hall, Inc., New York.

[16] D. Maier and J.D. Ullman, "Fragments Of Relations", *Proc. of the ACM SIGMOD Conference*, pp. 15-22, 1983.

[17] J.-P. Cheiney, P. Faudemay and R. Michel, "An Extension Access Paths To Improve Joins And Selections", *Proc. of the 1986 Int. Conf. on Data Engineering*, pp. 270-280, 1986.

[18] S. Ceri, M. Negri and G. Pelagatti, "Horizontal Data Partitioning In Database Design", *ACM Proceedings of SIGMOD '82*, pp. 128-136, 1982.

[19] T.K. Sellis, "Multiple-Query Optimization", *ACM Transactions on Database Systems*, Vol.13,No 1, pp.23-52, March 1988.