incompatibilities, for which there exist one-to-one mappings among the different databases. It does not deal with the problems of semantic incompatibilities, where there may not exist correspondences. However, the system will be considerably less expensive to build, simpler to implement, easier to maintain, and more flexible to fit the manufacturing environment. The system not only allows data to be automatically shared among the differing databases, but it also makes full use of the existing databases.

This research is only one step in the continuing process of building dynamic, flexible, real-time linkages between diverse, heterogenous CIM multidatabases [5]. Other research efforts under way are investigating different conceptual models for CIM databases, developing unique and expanded CIM control structures, and examining implementation issues when installing such a system. Issues such as the concurrency control, security, and data integrity are to be investigated in the future research.

References

[1] P. Chen, "The entity-relationship model—Toward a unified view of data," *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 9–36, 1976.

[2] U. Dayal and J. M. Smith, "PROBE: A knowledge-oriented database management systems," in *On Knowledge Base Management Systems*, M. L. Brodie and J. M. Smith, Eds. Berlin, Germany: Springer-Verlag, 1986, pp. 272–298.

[3] J. Davis and D. M. Oliff, "Requirements for the integration of manufacturing planning islands using knowledge based technology," in *Expert Systems and Intelligent Manufacturing*, D. M. Oliff, Ed. New York: Elsevier, 1988, pp. 29–42.

[4] D. M. Dilts, "Integration of computer integrated manufacturing data bases using artificial intelligence," in *Expert Systems and Intelligent Manufacturing*, M. Oliff, Ed. New York: Elsevier, 1988, pp. 327–334.

[5] D. M. Dilts and W. Wu, "Knowledge based systems for integrating computer integrated manufacturing data bases," in *Proc. Third Int. Conf. Expert Syst. Leading Edge Production Oper. Management*, South Carolina, May 1989, pp. 255–268.

[6] U. Flatau, "Designing an information system for integrated manufacturing systems," in *Design and Analysis of Integrated Manufacturing Systems*. Washington, DC: National Academy Press, 1988, pp. 60–78.

[7] R. Keller, *Expert System Technology: Development and Application*. Englewood Cliffs, NJ: Prentice-Hall, 1987.

[8] V. I. Kirov, "A CAD/CAPP expert system shell," in *Proc. Expert Sys. 85*, M. Merry, Ed. Cambridge, MA: Cambridge University Press, 1985, pp. 157–168.

[9] V. Kirshnamurthy, Y. W. Su, and H. Lam, "IMDAS—An integrated manufacturing data administration system," *Data Knowledge Eng.*, vol. 3, no. 4, pp. 109–131, 1988.

[10] J. F. Korsh and L. L. Garrett, *Data Structures, Algorithms, and Program Style Using C*. Boston, MA: PWS-KENT, 1988.

[11] F. Manola and M. L. Brodie, "On knowledge-based systems architectures," in *On Knowledge Base Management Systems*, M. L. Brodie and J. Mylopoulos, Eds. Berlin, Germany: Springer-Verlag, 1986, pp. 35–54.

[12] D. A. Marchand, "Strategic information management: Challenges and issues in the CIM environment," in *Expert Systems and Intelligent Manufacturing*, M. Oliff, Ed. New York: Elsevier, 1988, pp. 11–28.

[13] A. D. Meyer, "The integration of manufacturing information systems," in *Proc. 1988 Int. Conf. Comput. Integrated Manufacturing*, Troy, NY, May 27–29, 1988, IEEE Computer Society, 1988, pp. 247–259.

[14] T. Sellis, N. Roussopoulos, L. Mark, and C. Faloutsos, "Expert database systems: Efficient support for engineering environments," *Data Knowledge Eng.*, vol. 3, no. 4, pp. 71–85, 1988.

[15] V. A. Tipnis, "Process and economic models for manufacturing operations," in *Design and Analysis of Integrated Manufacturing Systems*. Washington, DC: National Academy of Engineering, 1988, pp. 92–117.

[16] W. Wu, D. M. Dilts, and S. Jiang, "A knowledge-based system for system dynamics modeling," in *Proc. 1989 Soc. for Comput Simulation Western Multiconf. Syst. Dynamics*, Simulation Council, Inc., 1989, pp. 35–40.

[17] J. A. White, "Material handling in integrated manufacturing systems," in *Design and Analysis of Integrated Manufacturing Systems*, W. D. Compton, Ed. Washington, DC: National Academy Press, 1988, pp. 46–59.

Reliability of Answers to Queries in Relational Databases

Fereidoon Sadri

Abstract— **We study the problem of determining the reliability of answers to queries in a relational database system, where the information in the database comes from various sources with varying degrees of reliability. An extended relational model is proposed in which each tuple in a relation is associated with an "information source vector" which identifies the information source(s) that contributed to that tuple. We show how relational algebra operations can be extended, and implemented using information source vectors, to calculate the vector corresponding to each tuple in the answer to a query, and hence, to identify information source(s) contributing to each tuple in the answer. This also enables the database system to calculate the "reliability" of each tuple in the answer to a query as a function of the reliabilities of information sources.**

Index Terms—**Databases, extended relational algebra, inaccurate information, query processing, reliability of answers.**

I. Introduction

The question of incomplete data in databases has received considerable attention (e.g., [1]–[5], to name a few). A closely related question, that of inaccurate data is relatively unexplored [6]. In this paper, we consider a model in which data come from various sources with known reliabilities, and study the problem of determining the reliability of answers to queries.

Our approach is to record, for each tuple in the database, the information source(s) of that tuple. This can simply be done by associating with each tuple the ID(s) of the information source(s), or, alternatively, to associate vectors with each tuple in a relation that determine the sources. In Section II, we introduce our model of extended relations, where each tuple is associated with a "source vector" that identifies information sources that contributed to that tuple. We discuss how relational algebra operations can be extended to derive the contributing information sources for each tuple in the answer to a query. Properties of these extended operations are discussed in Section III. We introduce the concept of equivalent (extended) relations, and show that certain identities of relational algebra hold for the extended operations under the notion of equivalence. In Section IV, we show how the probability of correctness of answers can be calculated from the reliability of information sources. Finally, in Section V, we present some discussions and further directions.

II. Source Determination for Derived Data

A. The Model

We associate vectors of length k, where k is the number of information sources, with each tuple in a relation to indicate the information sources that contributed, positively or negatively, to that tuple. For example, if the information source j confirmed the information reflected by a tuple t, then the jth element of the source vector associated with t is set to 1, and the remaining elements are 0. The source vectors associated with the tuples of the original database relations will usually have 1 and 0 elements, reflecting contributing (and noncontributing) information sources. Tuples in derived relations, such as answers to a query, might have source

Manuscript received March 17, 1989; revised May 29, 1990. This work was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

The author is with the Department of Computer Science, Concordia University, Montreal, P.Q., Canada, H3G 1M8.

IEEE Log Number 9144314.

vectors with -1, 0, and 1 elements. A source vector with a -1 entry for element j indicates that the corresponding tuple is valid if the information source j is incorrect. We will discuss source vectors in detail further below, and extend relational algebra operations to derive information sources of a tuple in the answer to a query.

Definitions: An *extended relation scheme* $R = \{A_1, \cdots, A_n, S\}$ is a set of attributes $A_1, \cdots, A_n$, S, where $A_1, \cdots, A_n$ are normal attributes, and S is a special attribute, called the *information source attribute* (*source attribute* for short). An *extended relation* (*instance*) r on the extended scheme R is a finite subset of $D_1 \times \cdots \times D_n \times D_s$, where $D_1, \cdots, D_n$ are domains of attributes $A_1, \cdots, A_n$, respectively, and D_s designates -1, 0, 1 k-vectors (i.e., vectors of length k with -1, 0, and 1 entries, where k is the number of information sources). We will often write a tuple in r as t @ v, where t is the projection of the tuple on $\{A_1, \cdots, A_n\}$ and is called the *pure tuple,* and v is the value of the tuple for the source attribute, and is called the *information source vector* (*source vector* for short). A pure tuple t in r can be associated with several source vectors, e.g., r can contain t @ $u_1, \cdots, t$ @ u_p. We use t @ x as a shorthand for t @ $u_1, \cdots, t$ @ u_p, where x is the set of source vectors $\{u_1, \cdots, u_p\}$. We will use u, v, and w for source vectors; and x, y, and z for sets of source vectors.

Interpretation of Source Vectors: We use the information source vectors to identify information sources that contributed to a tuple in an extended relation. The information source attribute and vectors are maintained and manipulated by the database system. Users do not have access to the source attribute and vectors. The only differences visible to the users are the following.

1) Users are asked to identity the information source(s) when they attempt to insert, delete, or modify a tuple.

2) The system can provide, for each tuple in the answer to a (user's) query, a list of contributing information sources.

3) If the users provide the database system with the *reliabilities* of information sources, then the system can calculate the reliability of each tuple in the answer to a query.

These features can be turned off, in which case the database system functions as a regular database system.

We represent information sources with variables $s_1, \cdots, s_k$, where k is the number of information sources. A source vector v (with -1, 0, and 1 entries) represents the conjunction of variables and negation of variables corresponding to the information sources with 1 and -1 entries, respectively. The interpretation of the same pure tuple with several source vectors, i.e., t @ $v_1, \cdots, t$ @ v_p, is the disjunction of clauses corresponding to $v_1, \cdots, v_p$.

Example 1: Consider a case where there are four information sources, which we will identify by variables s_1, s_2, s_3, and s_4. Assume the answer to a query consists of the following tuples:

$$t_1 \text{ @ } (1 \quad 0 \quad -1 \quad 0)$$
$$t_2 \text{ @ } (1 \quad -1 \quad 0 \quad 1)$$
$$t_2 \text{ @ } (0 \quad 0 \quad -1 \quad 1).$$

The first tuple indicates that t_1 is in the answer if source s_1 is correct and source s_3 is wrong. We will write this as $s_1 \wedge (\neg s_3)$.

Similarly the expression corresponding to t_2 is

$$(s_1 \wedge (\neg s_2) \wedge s_4) \vee (\neg s_3 \wedge s_4)$$

which indicates that t_2 is in the answer if either s_1 and s_4 are correct and s_2 is wrong, or s_4 is correct and s_3 is wrong.

We will discuss in Section IV how the reliability that a tuple t is in the answer to a query can be calculated as a function of the reliabilities of the information sources using the expression corresponding to t. □

B. Relational Algebra Operations

Now we are ready to explain how the operations should be extended in our model. First, we define the expression corresponding to a tuple, which is easily obtainable from the vectors associated with the tuple. Then we will explain how relational algebra operations should operate on these expressions, and also discuss how this can be implemented using source vectors.

Definition: Let R be an extended relation scheme. Let r be a relation instance on R, and let t be a pure tuple in r. In general, t corresponds to several extended tuples in r with the same pure component, e.g., t @ $v_1, \cdots, t$ @ v_p. With each information source j we associate a propositional variable s_j. The *expression corresponding to t,* denoted $e(t)$, is

$$e(v_1) \vee \cdots \vee e(v_p)$$

where $e(v_i)$, the expression corresponding to v_i, is the conjunction of all s_j where the vector v_i has a 1, and all $\neg s_j$ (negation of s_j) where the vector v_i has a -1. The expression corresponding to a pure tuple specifies the condition under which the pure tuple exists in terms of the propositional variables representing information sources.

Note that the above definition determines a unique expression for a (pure) tuple. We also know that there can be many other equivalent expressions. For example, the expression

$$(a \wedge b) \vee (a \wedge (\neg b))$$

is equivalent to the expression a. In our formulation this means that two tuples t @ $(1 \quad 1 \quad 0 \quad 0)$ and t @ $(1 \quad -1 \quad 0 \quad 0)$ can be replaced by the tuple t @ $(1 \quad 0 \quad 0 \quad 0)$. We are not concerned about this at the moment: our algorithm to calculate the reliability will generate the same answer for equivalent expressions (as it should!) identifying such equivalences, and replacing them by the equivalent (but shorter) representation can become important if we consider space requirements of this approach. We will discuss issues related to space consideration later.

The extended relational algebra operations take one (in the case of selection and projection) or two (in the case of Cartesian product, natural join, union, intersection, and set difference) extended relations as operands, and produce an extended relation as result. The operations are the same as the "classical" operations as far as pure tuples are concerned. It has been shown that selection, projection, Cartesian product, union, and set difference form a complete set of relational algebra operations. We have included natural join and intersection in the following discussion as these operations simplify query specification considerably. We will first discuss the manipulation of expressions corresponding to pure tuples, and then show how these manipulations can be implemented by source vectors in Section II-C.

Selection: Each pure tuple selected will retain its expression. The implementation in terms of source vectors is straightforward, i.e.,

$$\sigma_C(r) = \{t \text{ @ } x \in r \mid t \text{ satisfies condition } C\}$$

where x is the set of source vectors associated with t in r. Note that the special source attribute is hidden from users, and the condition C cannot refer to this attribute.

Projection: In this case, there is a possibility that several pure tuples in the relation project into the same pure tuple in the result. Assume pure tuples $t_1, \cdots, t_n$ in R project into t. Then

$$e(t) = e(t_1) \vee \cdots \vee e(t_n).$$

In other words, t exists in the answer if and only if t_1 or t_2 or $\cdots$ or t_n exist in r.

Union: Union is similar to projection. A pure tuple t in the answer may appear in one or both operands. The expression corresponding

to t in the answer is the disjunction of the expressions corresponding to t in the operands.

Cartesian Product, Intersection, and Natural Join: Consider $r \times s$. A pure tuple in the result, e.g., t, is due to two pure tuples t_1 and t_2 in r and s, respectively. The expression corresponding to t is the conjunction of the expressions corresponding to t_1 and t_2. This is due to the fact that "t exists if and only if both t_1 and t_2 exist."

$$r \times s = \{(t_1 \cdot t_2) \ @ \ x \mid t_1 \ @ \ y \in r, t_2 \ @ \ z \in s, e(t) = e(t_1) \wedge e(t_2)\}$$

where x, y, and z are sets of source vectors.

Intersection and natural join are similar to the Cartesian product.

Set Difference: Consider $r - s$. A pure tuple t exists in the result if and only if it exists in r, and does not exist in s. Hence, the expression of t in the result is the conjunction of the expression of t in r and the negation of the expression of t in s (if t appears in s).

$$e(t) = e_1(t) \wedge (\neg e_2(t))$$

where $e(t)$, $e_1(t)$, and $e_2(t)$ denote the expression corresponding to t in the result, in r, and in s, respectively.

We will discuss the implementation of extended relational algebra operations in the next section.

C. Source Vector Algorithms

In this section, we will describe how conjunction, disjunction, and negation of expressions can be implemented using our formulation of 1, 0, -1 vectors. The most general case corresponds to two pure tuples t_1 and t_2, contributing to one tuple in the answer, where t_1 @ $v_1, \cdots, t_1$ @ v_p are in the first relation, and t_2 @ $u_1, \cdots, t_2$ @ u_q are in the second relation.

Conjunction: In this case, we want to obtain a pure tuple t in the result of the form t @ $w_1, \cdots, t$ @ $w_g$where t is a function of t_1 and t_2, and $w_1, \cdots, w_g$ are functions of $v_1, \cdots, v_p$, and $u_1, \cdots, u_q$, such that

$$e(t) = e(t_1) \wedge e(t_2)$$

i.e., the expression associated with t is the conjunction of the expressions associated with t_1 and t_2. First consider the case with $p = q = 1$, i.e., t_1 @ $v$and t_2 @ u are the only tuples corresponding to pure tuples t_1 and t_2 in the two operands. Then there will be one tuple t @ w (i.e., $g = 1$) where w is obtained as follows:

Let $v = a_1 \cdots a_k$, and $u = b_1 \cdots b_k$ be nonzero vectors. If for some i, $a_i = 1$ and $b_i = -1$, or $a_i = -1$ and $b_i = 1$, then $w = 0$ (vector of all zeros), otherwise $w = c_1 \cdots c_k$ where c_i is derived using the following "truth" table. If $v = 0$, or $u = 0$, then $w = 0$. A zero vector corresponds to the truth value "false."

a_i	b_i	c_i
-1	-1	-1
-1	0	-1
-1	1	$w = 0$
0	-1	-1
0	0	0
0	1	1
1	-1	$w = 0$
1	0	1
1	1	1.

Note that this operation, restricted to 0, 1 vectors, corresponds to logical OR. Henceforth, we will call it 3OR (logical OR for three-valued vectors), and denote it by $\|$, i.e., $w = v\|u$.

For the general case, 3OR of two sets of source vectors $x = \{v_1, \cdots, v_p\}$ and $y = \{u_1, \cdots, u_q\}$ is calculated pairwise:

$$x\|y = \{v_1\|u_1, \cdots, v_1\|u_q, \cdots, v_p\|u_1, \cdots, v_p\|u_q\}.$$

Lemma 1: The following facts are direct consequences of the "truth" table for 3OR operation.

a) $v\|v = v$
b) $v_1\|v_2 = v_2\|v_1$
c) $v_1\|(v_2\|v_3) = (v_1\|v_2)\|v_3$.

Theorem 1: Let $e(t_1)$ and $e(t_2)$ be the expressions corresponding to t_1 and t_2 in t_1 @ x and t_2 @ y, where x and y are sets of source vectors. Let $e(t)$ be the expression corresponding to t in t @ z where $z = x\|y$, then $e(t) = e(t_1) \wedge e(t_2)$.

Proof: Let us first consider the case with singleton sets, i.e., $x = \{v_1\}$ and $y = \{u_1\}$. The source vector v_1 represents the expression $e(t_1)$ which is the conjunction of propositional variables s_i, for those i where the ith entry of v_1 is 1, and $\neg s_i$, for those i where the ith entry of v_1 is -1. Similarly, u_1 represents $e(t_2)$ in a similar manner. The conjunction of $e(t_1)$ and $e(t_2)$, $e(t_1) \wedge e(t_2)$, can be obtained by using the following rules from propositional logic:

1) $S \wedge S = S$
2) $S \wedge (\neg S) =$ false
3) $S \wedge T = T \wedge S$.

It can be easily verified that the "truth table" for 3OR executes the above rules to obtain $z = v_1\|u_1$.

Now for the general case, the 3OR operation is performed between $x = \{v_1, \cdots, v_p\}$ and $y = \{u_1, \cdots, u_q\}$ pairwise. This is a direct simulation of the propositional logic distribution rule

4) $(S_1 \vee S_2) \wedge (T_1 \vee T_2) = (S_1 \wedge T_1) \vee (S_1 \wedge T_2) \vee (S_2 \wedge T_1) \vee (S_2 \wedge T_2)$ □

Negation: This operation is needed for set difference. First consider t @ v, where v describes the expression of t, $e(t)$. We define an operation on source vectors, called NOT and denoted by # sign, such that $\#(v) = (u_1, \cdots, u_k)$ and the expression corresponding to t in t @ $u_1, \cdots, t$ @ u_k is the negation of the expression $e(t)$ in t @ v. Let $v = a_1 \cdots a_k$ be a nonzero source vector, then $\#(v) = (u_1, \cdots, u_k)$ where each u_i is a vector obtained from a_i as follows:

1) If $a_i = 1$ (-1) then u_i has its ith entry equal to -1 (1), and the rest 0.

2) If $a_i = 0$ then $u_i = 0$, and need not be included.

NOT of a set of vectors $x = \{v_1, \cdots, v_p\}$ is defined as the 3OR of NOT of each vector:

$$\#(\{v_1, \cdots, v_p\}) = \#(v_1)\|\cdots\|\#(v_p).$$

Lemma 2: The following fact is a direct consequence of the definition of the NOT operation:

$$\#(\#v) = v.$$

Theorem 2: Let $e(t)$ correspond to t in t @ x, where x is a set of source vectors, and let $e'(t)$ correspond to t @ y where $y = \#x$. Then $e'(t) = \neg e(t)$.

Proof: The proof follows from DeMorgan's laws of propositional logic. The negation of an expression of the form $e_1 \wedge e_2 \wedge \cdots \wedge e_k$ is $\neg e_1 \vee \neg e_2 \vee \cdots \vee \neg e_k$. This fact is used in the construction of $\#(v)$ for a single source vector v. Similarly, the negation of an expression of the form $e_1 \vee e_2 \vee \cdots \vee e_p$ is $\neg e_1 \wedge \neg e_2 \wedge \cdots \wedge \neg e_p$. This is exactly how $\#(\{v_1, \cdots, v_p\})$ is defined. □

Disjunction: Disjunction of expressions occurs for union and projection operations, whenever several tuples in the operands map to the same pure tuple in the result.

Theorem 3: Let $e_1(t)$ correspond to t @ x, and $e_2(t)$ correspond to t @ y, where x and y are sets of source vectors. Let $e(t)$ correspond to t @ z, where $z = x \cup y$. Then $e(t) = e_1(t) \vee e_2(t)$.

Proof: Follows from the definition of the expression corresponding to a pure tuple. □

Finally, we can give the relational algebra operations for extended relations. It is easy to show that these source vector operations implement the desired propositional logic operations on the corresponding expressions.

Selection, Projection, and Union:

Each tuple will retain its source vector, i.e.,

$$\sigma_C(r) = \{t \ @ \ x \in r \mid t \text{ satisfies condition } C\}$$

$$\Pi_X(r) = \{t[X] \ @ \ v \mid t \ @ \ v \in r\}$$

$$r \cup s = \{t \ @ \ v \mid t \ @ \ v \in r, \text{ or } t \ @ \ v \in s\}.$$

Intersection: The source vector for each tuple in the result is obtained as the 3OR of the source vectors of the underlying tuples, i.e.,

$$r \cap s = \{t \ @ \ (v_1 \| v_2) \mid t \ @ \ v_1 \in r \wedge t \ @ \ v_2 \in s\}.$$

Cartesian Product: The source vector for each tuple in the result is obtained as the 3OR of the source vectors of the underlying tuples, i.e.,

$$r \times s = \{(t_1 \cdot t_2) \ @ \ (v_1 \| v_2) \mid t_1 \ @ \ v_1 \in r \wedge t_2 \ @ \ v_2 \in s\}$$

where $t_1 \cdot t_2$ indicates concatenation of t_1 and t_2.

Natural Join:

$$r \bowtie s = \{(t_1 o t_2) \ @ \ (v_1 \| v_2) \mid t_1 \ @ \ v_1 \in r, t_2 \ @ \ v_2 \in s, \text{ and } t_1 \text{ and } t_2 \text{ join}\}$$

where $t_1 o t_2$ indicates the join of t_1 and t_2, i.e., the concatenation of t_1 and t_2 with the removal of duplicate values of common attributes.

Set Difference:

$$r - s = \{t \ @ \ x \mid t \ @ \ x \in r, \text{ and pure tuple } t \text{ is not in } s, \text{ or } t \ @ \ y \in r, \text{ and } t \ @ \ z \in s, \text{ and } x = y \| (\#z)\}.$$

Example 2: Given the relations SUPPLIER and PART, the query "List suppliers who only supply metal parts" is formulated and calculated as follows:

S #	P#	S			
s1	p1	1	0	0	0
s1	p2	1	0	0	0
s1	p3	1	0	0	0
s2	p2	0	1	0	0
s2	p4	1	0	0	0
s3	p3	0	1	0	0

SUPPLIER

P#	type	S			
p1	wood	0	0	1	0
p2	metal	0	1	0	0
p3	plastic	0	0	1	0
p4	metal	0	0	0	1

PART

$$\Pi_{S\#}(\text{SUPPLIER} \bowtie (\sigma_{C1}(\text{PART}))) - \Pi_{S\#}(\text{SUPPLIER} \bowtie (\sigma_{C2}(\text{PART})))$$

where $C1$ = (type = "metal"), and $C2$ = (type $\neq$ "metal"). We have shown below how the answer is calculated in steps:

$$r_1 = \sigma_{C1}(\text{PART})$$

$$r_2 = \sigma_{C2}(\text{PART})$$

$$r_3 = \text{SUPPLIER} \bowtie r_1$$

$$r_4 = \text{SUPPLIER} \bowtie r_2$$

$$r_5 = \Pi_{S\#}(r_3)$$

$$r_6 = \Pi_{S\#}(r_4)$$

$$r_7 = r_5 - r_6$$

P#	type	S			
p2	metal	0	1	0	0
p4	metal	0	0	0	1

r_1

P#	type	S			
p1	wood	0	0	1	0
p3	plastic	0	0	1	0

r_2

S#	P#	type	S			
s1	p2	metal	1	1	0	0
s2	p2	metal	0	1	0	0
s2	p4	metal	1	0	0	1

r_3

S#	P#	type	S			
s1	p1	wood	1	0	1	0
s1	p3	plastic	1	0	1	0
s3	p3	plastic	0	1	1	0

r_4

S#	S			
s1	1	1	0	0
s2	0	1	0	0
s2	1	0	0	1

r_5

S#	S			
s1	1	0	1	0
s3	0	1	1	0

r_6

S#	S			
s1	1	1	−1	0
s2	0	1	0	0
s2	1	0	0	1

r_7

Let the four information sources be represented by A, B, C, and D. The answer shows that supplier s_1 supplies only metal parts if sources A and B are correct, and source C is wrong; and s_2 supplies only metal parts if either B is correct, or both A and D are correct. □

III. Properties of Extended Relational Algebra Operations

In the previous section we extended the relational algebra operations to manipulate source vectors according to the intended semantics of the operations. In this section, we will investigate whether some well known properties of relational algebra also hold for the extended operations. We introduce the concept of "equivalent" extended relations, and show that certain identities for relational algebra hold for the extended operations under the notion of equivalence.

Definition: Let $x = \{v_1, \cdots, v_p\}$ and $y = \{u_1, \cdots, u_q\}$ be two sets of source vectors. We say x and y are *equivalent*, written $x \equiv y$, if the expressions corresponding to x and y are logically equivalent. For example, the following are equivalent:

$$x = \{(1 \quad 0 \quad 0 \quad 0)\}, \quad \text{and}$$

$$y = \{(1 \quad -1 \quad 0 \quad 0), (1 \quad 1 \quad 0 \quad 0)\}.$$

Lemma 3: Let x and y be sets of source vectors. Then

a) $(x \cup y) \| z \equiv (x \| z) \cup (y \| z)$
b) $(x \| y) \cup z \equiv (u \cup z) \| (y \cup z)$
c) $\#(x \cup y) \equiv \#x \| \#y$
d) $\#(x \| y) \equiv (\#x \cup \#y)$.

Proof: We know from Theorems 1, 2, and 3 that the source vector operations $\cup$, $\|$, and # correspond to the logical operations of disjunction, conjunction, and negation. The above equivalences follow from logical distribution rules, and DeMorgan's rules. □

Definition: Two extended relations r and s are *equivalent*, written $r \equiv s$, if for all pure tuples t if x and y are the sets of source vectors associated with t in r and s, respectively, then $x \equiv y$.

Definition: A source vector u *covers* a source vector v if for all j, if $u[j] \neq 0$, then $u[j] = v[j]$, where $u[j]$ and $v[j]$ denote the jth element of u and v, respectively. For example,

$$u = (-1 \quad 0 \quad 0 \quad 1 \quad 0) \text{ covers } v = (-1 \quad 0 \quad 1 \quad 1 \quad -1).$$

Lemma 4: Let u and v be two source vectors, and let u cover v. Then $e(u) \vee e(v) = e(u)$, where $e(u)$ is the expression associated with u.

Proof: It is easy to verify that the ones of $e(v)$ is covered by the ones of $e(u)$ (for example in a Carnaugh map). □

Lemma 5: For an extended relation r,

a) $r \cup r = r$,

b) $r \cap r \equiv r$.

Proof: a) follows from the definition of the (extended) union operation. In case of b) we can write

$$r \cap r = \{t \mathbin{@} (x \| x) \mid t \mathbin{@} x \in R\}$$

where x is the set of source vectors associated with the pure tuple t in r. Let $x = \{u_1, \cdots, u_k\}$. Then $x \| x$ is calculated pairwise, i.e.,

$$x \| x = \{u_1 \| u_1, \cdots, u_1 \| u_k, \cdots, u_k \| u_1, \cdots, u_k \| u_k\}$$

by Lemma 1 we know $u_i \| u_i = u_i$. We can easily verify that u_i covers $u_i \| u_j$ for all j. It follows from Lemma 4 that $x \| x \equiv x$, and hence b) is proved. □

Lemma 6: For the extended relations r and s

$$r \cap s \equiv r - (r - s)$$

where the (extended) intersection and set difference are as defined in Section II.

Proof: We notice that

$$r - s = \{t \mathbin{@} x \mid t \mathbin{@} x \in r, \text{ and pure tuple } t \text{ is not in } s, \text{ or } t \mathbin{@} y \in r, \text{ and } t \mathbin{@} z \in s, \text{ and } x = y \| (\#z)\}$$

then

$$r - (r - s) = \{t \mathbin{@} x \mid t \mathbin{@} y \in r, \text{ and } t \mathbin{@} z \in s, \text{ and } x = y \| (\#(y \| (\#z)))\}$$

but, using Lemmas 2 and 3,

$$y \| (\#(y \| (\#z))) \equiv y \| (\#y \cup \#(\#z)) \equiv y \| (\#y \cup z) \equiv y \| z.$$

Then

$$r - (r - s) \equiv \{t \mathbin{@} x \mid t \mathbin{@} y \in r, \text{ and } t \mathbin{@} z \in s, \text{ and } x = y \| z\} \equiv r \cap s.$$

Lemma 7: Let r and s be extended relations on schemes R and S, and let $R \cup S = V$ and $R \cap S = W$, then

$$r \bowtie s = \Pi_V(\sigma_{r.W = s.W}(r \times s)).$$

Proof: We note that

$$q = \sigma_{r.W = s.W}(r \times s) = \{(t_1 \cdot t_2) \mathbin{@} (x \| y) \mid t_1 \mathbin{@} x \in r, \; t_2 \mathbin{@} y \in s, \text{ and } t_1[W] = t_2[W]\}.$$

Also, each tuple in the projection of q onto V results from exactly one tuple in q (otherwise, q would have duplicate tuples), hence

$$\Pi_V(\sigma_{r.W = s.W}(r \times s)) = \{(t_1 \cdot t_2)[V] \mathbin{@} (x \| y) \mid t_1 \mathbin{@} x \in r, \; t_2 \mathbin{@} y \in s, \text{ and } t_1[W] = t_2[W]\}$$

which is the definition of the natural join. □

IV. Reliability of Answers to a Query

If we know the reliability (i.e., the probability of being correct) of information sources, we can calculate the reliability of answers to a query.

Definition: The reliability of a source is defined as the probability that a tuple coming from that source is valid. We designate the reliability of source j by $\mathrm{re}(j)$. We assume that different information sources are independent.

Consider a tuple $t \mathbin{@} v$ in the result of a query, where v is a source vector. We would like to calculate the reliability of t, i.e., the probability that t is an answer, from the reliability of the sources. Let the elements of v with a value of 1 correspond to sources $i_1, \cdots, i_p$, that is, $v[i_1] = \cdots = v[i_p] = 1$, where $v[i]$ designates the ith element of v. Similarly, let the elements of v with a value of -1 correspond to sources $j_1, \cdots, j_q$. The reliability of the tuple $t \mathbin{@} v$, written $\mathrm{rel}(t)$, is calculated as

$$\mathrm{rel}(t) = \mathrm{re}(i_1) \times \cdots \times \mathrm{re}(i_p) \times (1 - \mathrm{re}(j_1)) \times \cdots \times (1 - \mathrm{re}(j_q)).$$

That is, the probability that t is valid is the product of the probabilities that sources $i_1, \cdots, i_p$ are correct, and sources $j_1, \cdots, j_q$ are wrong. Note that the assumption that information sources are independent is used in this formula.

We will need a definition before proceeding to other operations:

Definition: Two source vectors v_1 and v_2 are said to be *independent* if for no source j both of them have a nonzero entry. A set of source vectors is *independent* if they are pairwise independent.

The reliability of a tuple $t \mathbin{@} x$, where $x = \{v_1, \cdots, v_p\}$ is the set of source vectors associated with the pure tuple t, can be calculated as follows if $\{v_1, \cdots, v_p\}$ are independent.

$$\mathrm{rel}(t) = 1 - (1 - \mathrm{rel}(t \mathbin{@} v_1)) \times \cdots \times (1 - \mathrm{rel}(t \mathbin{@} v_p))$$

where $\mathrm{rel}(t \mathbin{@} v_i)$is the reliability of t when only $t \mathbin{@} v_i$ is considered. The case for interdependent source vectors will be discussed later.

Example 3: In Example 2 we found the answer to the query "list suppliers who supply only metal parts", repeated below for convenience

$S\#$	S
$s1$	1 1 −1 0
$s2$	0 1 0 0
$s2$	1 0 0 1

Now we will calculate the reliabilities: Reliability that supplier s_1 is an answer:

$$\mathrm{rel}(s_1) = \mathrm{re}(1) \times \mathrm{re}(2) \times (1 - \mathrm{re}(3)).$$

For supplier s_2 there are two tuples. But the two tuples are independent:

$$\mathrm{rel}(s_2 \mathbin{@} 0100) = \mathrm{re}(2)$$

$$\mathrm{rel}(s_2 \mathbin{@} 1001) = \mathrm{re}(1) \times \mathrm{re}(4).$$

Then

$$\mathrm{rel}(s_2) = 1 - [(1 - \mathrm{re}(2)) \times (1 - \mathrm{re}(1) \times \mathrm{re}(4))].$$

For example, if the reliability of information sources are %90, %80, %70, and %60, respectively, then

$$\mathrm{rel}(s_1) = \%21.6$$

$$\mathrm{rel}(s_2) = \%90.8.$$

Note that if the sources were all %100 reliable, we would get

$$\text{rel}(s_1) = 0$$
$$\text{rel}(s_2) = \%100$$

which is consistent with the answer to the query in the "classical" relational model.

Reliability Calculation Algorithms: The remaining case to be considered is the reliability of t in $t \ @ \ v_1, \cdots, t \ @ \ v_p$, where $\{v_1, \cdots, v_p\}$ are not independent. We will describe two algorithms. The first one is based on the principle of inclusion and exclusion [7], and the second one uses expansion of Boolean expressions.

Algorithm 1: Let $t \ @ \ v_1, \cdots, t \ @ \ v_p$ be all the tuples corresponding to the pure tuple t. Let

$$K1 = \sum_{i=1}^{p} \text{re}(v_i)$$
$$K2 = \sum_{i=1}^{p} \sum_{j>i}^{p} \text{re}(v_i \| v_j)$$
$$K3 = \sum_{i=1}^{p} \sum_{j>i}^{p} \sum_{k>j}^{p} \text{re}(v_i \| v_j \| v_k)$$
$$\cdots.$$

That is, $K1$ is the sum of reliabilities of the vectors, $K2$ is the sum of reliabilities of pairwise 3OR of the vectors, etc., then

$$\text{rel}(t) = K1 - K2 + K3 - K4 + \cdots.$$

Algorithm 2: Let $t \ @ \ v_1, \cdots, t \ @ \ v_p$ be all the tuples corresponding to the pure tuple t. The idea is to find an equivalent set $t \ @ \ u_1, \cdots, t \ @ \ u_q$ where the vectors $u_1, \cdots, u_q$ correspond to an expression in disjunctive normal form, and then calculate the sum of their reliabilities. To obtain the equivalent set $t \ @ \ u_1, \cdots, t \ @ \ u_q$ we proceed as follows.

First, we do not need to consider sources that do not contribute to any of the u_i's. This may result in considerable improvement in the efficiency of the algorithm. We say a source s *contributes* to t in $t \ @ \ v_1, \cdots, t \ @ \ v_p$ if at least one v_i, $i = 1, \cdots, p$, has a nonzero entry for s. Let C denote the set of sources contributing to t. A vector u is said to be in standard form if all entries corresponding to sources in C are nonzero. Given a vector v, we can form an equivalent set of vectors $W = \{w_1, \cdots, w_l\}$, where each w_i is in standard form, by including all 1, -1 combinations of contributing sources that are 0 in v. For example, if $v = (1\ 0 - 1\ 0\ 0)$ and contributing sources are 1, 2, 3, and 5, then

$$W = \{(1 \quad 1 - 1 \quad 0 \quad 1), (1 \quad 1 - 1 \quad 0 - 1),$$
$$(1 - 1 - 1 \quad 0 \quad 1), (1 - 1 - 1 \quad 0 - 1)\}.$$

Given $t \ @ \ v_1, \cdots, t \ @ \ v_p$, the equivalent tuples $t \ @ \ u_1, \cdots, t \ @ \ u_q$, where all u_i's are in standard form, can be obtained by replacing each v_i, $i = 1$ to p, by its equivalent standard form set, and eliminating duplicates (this is the important point: by eliminating duplicates we are making sure that the calculated reliability is correct).

Finally, if $t \ @ \ w_1, \cdots, t \ @ \ w_l$ is an equivalent set to $t \ @ \ v_1, \cdots, t \ @ \ v_p$ where all w_i's are in standard form, then

$$\text{rel}(t) = \text{re}(w_1) + \cdots + \text{re}(w_l).$$

Example 4: Let the following tuples correspond to the pure tuple t.

$$t \ @ 1 \quad 0 \quad 1$$
$$t \ @ 1 \quad 1 \quad 0.$$

Then, by Algorithm 1:

$$K1 = \text{re}(1) \times \text{re}(3) + \text{re}(1) \times \text{re}(2)$$
$$K2 = \text{re}(1) \times \text{re}(2) \times \text{re}(3)$$
$$\text{rel}(t) = \text{re}(1) \times \text{re}(3) + \text{re}(1) \times \text{re}(2) - \text{re}(1) \times \text{re}(2) \times \text{re}(3).$$

Algorithm 2 first finds the equivalent vectors in standard form:

$$t \ @ \ 1 \quad 1 \quad 1$$
$$t \ @ \ 1 \quad -1 \quad 1$$
$$t \ @ \ 1 \quad 1 \quad -1$$

Then

$$\text{rel}(t) = \text{re}(1) \times \text{re}(2) \times \text{re}(3) + \text{re}(1) \times (1 - \text{re}(2)) \times \text{re}(3) + \text{re}(1) \times \text{re}(2) \times (1 - \text{re}(3))$$

which can be simplified to

$$\text{rel}(t) = \text{re}(1) \times \text{re}(3) + \text{re}(1) \times \text{re}(2) - \text{re}(1) \times \text{re}(2) \times \text{re}(3).$$

V. Discussion

We presented an approach to calculate reliability of answers to a query in a database where information comes from sources of different reliabilities. Our approach is based on two important assumptions:

1) Reliabilities are associated with sources of information.

2) Tuples are considered as the unit of data, and source information is recorded for each tuple.

We have intentionally avoided the definition of "source." In fact, the model is quite general in the sense that different choices for "sources" will result in different approaches to the correctness of query answers. A possible choice for "source" can take into account 1) where the data are coming from, 2) the topic of data, 3) reliability of the communication channel, 4) time when the data were obtained, and communicated, etc.

It can be argued that the choice of tuples as the unit of data is not realistic [8]. Consider for example a relation that represents an entity set: a single tuple may contain many attributes of an entity, which may be input at different times and may come from different sources. Similarly, a tuple in a relation that represents a relationship among several entities may reflect multiple facts confirmed by different sources. A possible approach to remedy this problem is to use a decomposition into binary relations. Another approach, which needs further investigation, is to use the formulation of [8] and associate sources with "facts" rather than with tuples.

A straightforward implementation of our approach, with source vectors of $2k$ bits, where k is the number of information sources, would result in poor space utilization when k becomes large. We observe that source vectors are very sparse, even for derived relations. An encoding scheme can be used to represent source vectors, and the algorithms can be implemented in terms of the encoding. The simplest approach is to represent a source vector by two lists, $\{i_1, \cdots, i_p\}$ $\{j_1, \cdots, j_q\}$, where the first list indicates $+1$ entries, and the second list indicates -1 entries. For most of the tuples in base relations there is only one member in the first list, and the second list is empty. The lists can be kept in sorted order to increase the efficiency of the algorithms represented in this paper.

Acknowledgment

The author wishes to thank F. Blanchet-Sadri for helpful discussions, as well as anonymous referees whose comments and suggestions helped to improve the paper substantially.

REFERENCES

[1] A. M. Keller and M. Winslett-Wilkins, "On the use of an extended relational model to handle changing incomplete information," *IEEE Trans. Software Eng.*, vol. SE-11, no. 7, pp. 620–633, July 1985.

[2] M. Winslett, "Updating logical databases with null values," in *Proc. 1st Int. Conf. Database Theory,* Rome, Sept. 1986.

[3] T. Imielinski and W. Lipski, "On representing incomplete information in a relational database," in *Proc. 7th Int. Conf. Very Large Databases,* 1981.

[4] W. Lipski, "On semantic issues connected with incomplete information databases," *ACM Trans. Database Syst.,* vol. 4, no. 3, pp. 262–296, Sept. 1979.

[5] J. Biskup, "A foundation of Codd's relational maybe operations," *ACM Trans. Database Syst.,* vol. 8, no. 4, pp. 608–636, 1983.

[6] K-C. Liu and R. Sunderraman, "On representing indefinite and maybe information in relational databases," in *Proc. IEEE Fourth Int. Conf. Data Eng.,* pp. 250–257, Feb. 1988.

[7] C. L. Lu, *Introduction to Combinatorial Mathematics.* New York: McGraw-Hill, 1968, ch. 4.

[8] B. C. Desai, P. Goyal, and F. Sadri, "Fact structure and its application to updates in relational databases," *Inform. Syst.,* vol. 12, no. 2, pp. 215–221, 1987.