

RELATIONAL STRATEGIES FOR PROCESSING UNIVERSALLY QUANTIFIED QUERIES TO LARGE DATA BASES

Koichi Furukawa
Information Systems Section
Computer Science Division
Electrotechnical Laboratory
Tokyo, Japan

Relational algebraic strategies are described for processing universally quantified queries. First, a new algorithm is developed to evaluate universally quantified conditional queries of the form $[\vec{x} : (\forall \vec{y}) W1(\vec{x}, \vec{y}) \rightarrow W2(\vec{x}, \vec{y})]$. It decomposes the problems into two subproblems using set intersection, summary and join operations within the frame work of relational algebra. Second, the partial multi-valued-dependency decomposition of a relation is introduced as an efficient representation schema to express a set of uniform intensional data of the form $(\forall \vec{x}/\vec{t})P(\vec{x})$. The original relation is defined in terms of the decomposed relations as a relational algebraic equation. A relational algebraic expression corresponding to a query to the original relation is transformed to the one in terms of the decomposed relations using the above equation. Furthermore, the transformed algebraic expression is converted to a simple associative search expression to one of the decomposed relations by applying the rule-based formula manipulation or symbolic evaluation method.

1. INTRODUCTION

It has been shown in Chang [1978], Furukawa [1977], Minker [1978], Kellogg et al. [1978] and Reiter [1978b] that deductive logic offers considerable potential for improving on-line access to large, complex data base domains. The common feature of the current researches to that direction is the attempt to combine a deductive component with relational data bases.

Chang [1978] divided the approaches of these researches into two groups: One is the evaluational approach where intensions are used to transform queries and extensions are used to evaluate queries. The other is the non-evaluational one where both intensions and extensions are used to prove a question represented by a formula in the same manner. It has been shown in Reiter [1978b] that the evaluational approach is more feasible for data bases with very large extensions and comparatively small intensions.

Codd [1972] has defined an algorithm to convert queries expressed in relational calculus to a sequence of relational algebraic operations. Reiter [1978b] has developed a general framework in the restricted first order logic which enables one to get indefinite answers to any

queries, and reformulated Codd's algorithm in his framework. Reiter [1978a] has also developed an efficient query conversion algorithm under the closed world assumption (CWA).

In this paper, a new algorithm to evaluate universally quantified conditional queries under the CWA is presented. Summary operation on a relation is added to the set of primitive operations of the relational algebra. The algorithm reduces the problem to the application of the operations of set intersection, summary and join to the answers to the subqueries.

In deductive relational data bases developed so far, only extensions have been considered to compose relations. Intensions have been stored in knowledge bases outside the relational data bases. However, it sometimes happens that we have a set of uniform intensional data of the form $(\forall \vec{x}/\vec{t})P(\vec{x})$. In this paper, a new representation schema based on the partial multi-valued dependency (PMVD) decomposition of a relation is introduced as a suitable way to express such data.

Queries to the decomposed relations are evaluated using a set of equations which give users the image of the original relations as a logical view. Furthermore, it will be shown that

universally quantified queries on those relations can be evaluated very efficiently by applying the rule-based symbolic evaluation method.

2. REVIEW OF QUERY EVALUATION UNDER THE CWA

In this section, query evaluation under the CWA developed by Reiter [1978a] will be reviewed quickly as well as some extensions.

All queries have the form $[x_1/\tau_1, \dots, x_n/\tau_n : (q_1 y_1/\theta_1) \dots (q_m y_m/\theta_m) W(x_1, \dots, x_n, y_1, \dots, y_m)]^n$ where $W(x_1, \dots, x_n, y_1, \dots, y_m)$ is a quantifier free formula with free variables $x_1, \dots, x_n, y_1, \dots, y_m$ and q_i is either $\forall$ or $\exists$, $i = 1, \dots, m$. We shall use the abbreviated notation $[x/\tau : (q y/\theta) W(x, y)]$ to express a typical query. The τ 's and θ 's, which are called types, are sets of constant signs which the variables associated with them range over. A sequence of types $\vec{\tau} = (\tau_1, \dots, \tau_n)$ is the set $\tau_1 x_1 \dots \tau_n x_n$.

A data base (DB) is a set of clauses containing no functional signs.

Under the CWA, if no proof of a positive ground literal exists, then the negation of that literal is assumed true. This can be viewed as equivalent to implicitly augmenting the given data base with all such negated literals.

Since worlds are completely specified under the CWA, the set of CWA answers $IQ!^{CWA}$ to $Q = [x/\tau : (q y/\theta) W(x, y)]$ is defined as follows:

$$IQ!^{CWA} = \{\vec{c} : \vec{c} \in \vec{\tau} \text{ and } DB \cup \overline{EDB} \vdash (q y/\theta) W(x, y)\}$$

where $\overline{EDB} = \{\bar{P}c : P \text{ is a predicate sign, } \vec{c} \text{ a tuple of constant signs and } DB \vdash P\vec{c}\}$.

The evaluation of any queries are reduced to the applications of the relational algebraic operations to the answers to atomic queries. Before listing query conversion rules, we define some of the operations of relational algebra.

Let $Q = [x/\tau, z/\psi : W]$, and $\vec{x}$ is the n-tuple $x_1, \dots, x_n$. Then $IQ!$ is a set of (n+1)-tuples, and the projection of $IQ!$ with respect to z , $\pi IQ!$ is the set of n-tuples obtained from $IQ!$ by deleting the (n+1)st component from each (n+1)-tuple of $IQ!$.

Let $Q = [x/\tau, z/\psi : W]$. Then the quotient of $IQ!$ by z , $\Delta_z IQ!$ is a set of tuples and is defined as follows:

$$\vec{c} \in \Delta_z IQ! \text{ iff } (\vec{c}, a) \in IQ! \text{ for all } a \in \psi.$$

The operator Δ_z is called the division with respect to z.

Let $Q1 = [x/\tau, z/\psi : W1]$ and $Q2 = [y/\theta, z/\psi : W2]$. The join of $IQ1!$ and $IQ2!$ with respect to z , $*_z(IQ1!, IQ2!)$, is a set of tuples and defined as follows:

$$(\vec{c}, d, \vec{e}) \in *_z(IQ1!, IQ2!)$$

$$\text{iff } (\vec{c}, d) \in IQ1! \text{ and } (\vec{e}, d) \in IQ2!.$$

We sometimes denote $*_z(IQ1!, IQ2!)$ as $(IQ1! *_z IQ2!)$.

The set of query conversion rules are listed below:

Rule 1. (Decomposition of AND/OR queries)

1. $! [x/\tau : W1 \wedge W2]! = ! [x/\tau : W1]! \cap ! [x/\tau : W2]!$
2. $! [x/\tau : W1 \vee W2]! = ! [x/\tau : W1]! \cup ! [x/\tau : W2]!$

Rule 2. (Elimination of negation)

1. $! [x/\tau : \bar{W}]! = \vec{c} - ! [x/\tau : W]!$
2. $! [x/\tau : W1 \wedge \bar{W2}]! = ! [x/\tau : W1]! - ! [x/\tau : W2]!$

Rule 3. (Elimination of existential quantifiers)

$$! [x/\tau : (\exists y/\theta) W]! = \pi_{y1} ! [x/\tau, y/\theta : W]!$$

where π_{y1} denotes $\pi_{y1} \dots \pi_{ym}$ and π_{y1} is the projection with respect to $y1$.

Rule 4. (Elimination of universal quantifiers)

$$! [x/\tau : (\forall y/\theta) W]! = \Delta_{y1} ! [x/\tau, y/\theta : W]!$$

where Δ_{y1} denotes $\Delta_{y1} \dots \Delta_{ym}$ and Δ_{y1} is the division with respect to $y1$.

Rule 5. (Decomposition of AND queries)

$$! [x/\tau : W1 \wedge W2]! = (! [x1/\tau1 : W1]! *_z ! [x2/\tau2 : W2]!)$$

where $x1$ is a sub-sequence of x which appears in $W1$, and $\tau1$ is the corresponding sub-sequence of τ , for $i = 1, 2$. The join operation $*$ is performed with respect to the variables included in both $x1$ and $x2$.

Most of the above rules are derived from Reiter [1978a]. The extended points are as follows:

1. In rule 2, the restriction that $W, W1$ and $W2$ must be quantifier free is removed.

2. Rule 4 is new. Reiter has developed a general framework which enables one to get indefinite answers. The division operator defined by Codd [1972] has been properly generalized in the framework. However, the handling of universal quantifiers in the CWA has been omitted in Reiter [1978a]. But there is no reason to exclude the handling of universal quantifiers in the CWA and moreover the set of conversion rules becomes more well-structured by adding these rules.

3. Rule 5 is new, too. It is easily shown that

this rule reduces the amount of computation. Let $Q_1 = [\bar{x}/\bar{z}:W_1]$ and $\bar{z}_1^0 = \pi_{\bar{x}_1}(\bar{z})$, i.e., the compliment subsequence of $\bar{z}$, $i = 1, 2$. We would have to evaluate $Q = [\bar{x}/\bar{z}:W_1 \wedge W_2]$ according to $!Q! = !Q_1! \wedge !Q_2!$, unless we have Rule 5. But it is easily shown that

$$!Q_1! = \bar{z}_1^0 \times ![\bar{x}_1/\bar{z}_1:W_1]!$$

for $i = 1, 2$. By applying Rule 5, we can avoid the execution of the above costly Cartesian product operations.

3. THE HANDLING OF UNIVERSAL QUANTIFIERS

Let us consider how to compute the division $\Delta_z!Q!$ where $Q = [\bar{x}/\bar{z}, z/\psi : (q\bar{y}/\bar{\theta})W]$. The following algorithm generates the answer $\Delta_z!Q!$.

Algorithm D.

1. Group the answer set $!Q!$ to the query Q by $\bar{x}$ and make a list of z 's for each $\bar{x}$ (we denote the list by $Z\bar{x}$).
2. For each $Z\bar{x}$, test whether it includes the set ψ . If the test succeeds, then put the $\bar{x}$ in the answer set $\Delta_z!Q!$.

Rule 4 says that $\Delta_z!Q!$ is the answer set for the universally quantified query $Q_0 = [\bar{x}/\bar{z} : (\forall z/\psi)(q\bar{y}/\bar{\theta})W]$. Notice that Q_0 is the abbreviation of the conditional query $Q_0' = [\bar{x}/\bar{z} : (\forall z)(z\psi \rightarrow (q\bar{y}/\bar{\theta})W)]$. The inclusion test $\psi \subset Z\bar{x}$ in Algorithm D reflects the implication in Q_0' , where the consequent $z\psi$ in Q_0' corresponds to the lefthand set ψ and the antecedent $(q\bar{y}/\bar{\theta})W$ to the righthand set $Z\bar{x}$.

We can use the division operation to efficiently evaluate a class of conditional queries $Q_c = [\bar{x}/\bar{z} : (\forall z/\psi)(W_1 \rightarrow W_2)]$ such that W_1 does not contain any free variables in $\bar{x}$. The following theorem states the fact more precisely.

Theorem 1. Let $Q_c = [\bar{x}/\bar{z} : (\forall z/\psi)(W_1 \rightarrow W_2)]$ be a universally quantified conditional query and assume that W_1 does not contain any free variables in $\bar{x}$, $Q = [\bar{x}/\bar{z}, z/\psi : W_2]$, and $Q_z = [z/\psi : W_1]$. If $DB \cup EDB$ is consistent, then

$$!Q_c! = \Delta_{z' \in !Q_z!} !Q! \text{ if } !Q_z! \neq \emptyset, \\ = \bar{z} \text{ otherwise.}$$

We sometimes denote $\Delta_{z' \in !Q_z!} !Q!$ as $!Q! / !Q_z!$.

We cannot use Theorem 1 if W_1 contains any variables in $\bar{x}$. We can deal with the general case by introducing the following cut operation:

Let $R = ![\bar{x}/\bar{z}, z/\psi : W(\bar{x}, z)]!$. Then the cut of R

with respect to $\bar{x}$, denoted by $\rho_{\bar{x}}(R)$, is a set of pairs $(\bar{c}_1, D_1)$ where $\bar{c}_1 \in \bar{z}$ and D_1 is a set of $d_k \in \psi$ such that

$$D_1 = \{d_k : (\bar{c}_1, d_k) \in R\}.$$

We regard $\rho_{\bar{x}}(R)$ to be a function from $\pi_{\bar{z}}R$ to $2^{\pi_{\bar{z}}R}$ and denote D_1 as $\rho_{\bar{x}}(R)(\bar{c}_1)$. $\rho_{\bar{x}}(R)(\bar{c}_1)$ is called the cross-section of $\rho_{\bar{x}}(R)$ at $\bar{c}_1$.

The set of D is exactly the same as $Z\bar{c}_1$ in Algorithm D.

Theorem 2. Let $Q = [\bar{x}/\bar{z} : (\forall z/\psi)(W_1 \rightarrow W_2)]$. If $DB \cup EDB$ is consistent, then

$$!Q! = (\bar{z} - ![\bar{x}/\bar{z} : (\exists z/\psi)W_1]!) \cup \{ \bar{c} : \phi \neq \rho_{\bar{x}_1}(R_1)(\bar{c}_1) \subset \rho_{\bar{x}_2}(R_2)(\bar{c}_2) \},$$

where $R_i = ![\bar{x}_i/\bar{c}_i, z/\psi : W_i]!$ for $i = 1, 2$.

Since $A \subset B$ iff $|A \cap B| = |A|$ in case A and B are finite sets, we can replace the set inclusion test in Theorem 2 by a simple comparison of the cardinal numbers of two sets. This strategy has been implemented in GM-RDMS (Whitney [1974]), where an operation called summary has been used to construct a set of pairs $(\bar{c}_1, |D_1|)$ which summarize the pairs $(\bar{c}_1, D_1)$.

Example

The following example is taken from Palermo [1974]. There are four relations: SUPPLY, SUPPLIERS, PROJECT and PART, as shown in Figure 1 (The relation names are underlined to distinguish them from the corresponding predicates. They are considered to refer sets, rather than predicates).

<u>SUPPLY</u>	SID	PID	JID	<u>SUPPLIER</u>	SID	SLOC	SNAME
	211	31	971		211	NY	AA
	325	32	971		325	SF	XX
	211	33	970		237	LA	YY
	211	31	972				
	237	31	972				
	237	31	970	<u>PROJECT</u>	JID	JLOC	JNAME
	237	32	970				
	237	33	970		970	POK	A
	237	32	971		971	SJ	X
	237	31	971		972	SJ	Y
<u>PART</u>	PID	PTYPE					
	31	A					
	32	A					
	33	B					

Figure 1. An example data base.

The relation SUPPLY (SID, PID, JID) is a set of tuples (x, y, z) such that a supplier x supplies a part y to a project z. SUPPLIER (SID, SLOC, SNAME) has information on suppliers' locations and their names. PROJECT (JID, JLOC, JNAME) has information on projects' locations and their names. PART (PID, PTYPE) defines the type of each part.

Let us consider the following query to the above data base:

Query. Find the name and location of suppliers each of whom has supplied at least one project located in San Jose with at least one of every part of type A.

This query is expressed as follows:

$Q = [x/SNAME, y/SLOC : (\exists s/SID)(SUPPLIER(s,y,x) \wedge (\exists j/JID)(PROJECT(SJ,j) \wedge (\forall p/PID)(PART(p,A) \rightarrow SUPPLY(s,p,j))))]$

The evaluation of Q proceeds as follows:

1. Let
 $Q1 = [x/SNAME, y/SLOC, s/SID : SUPPLIER(s,y,x)]$
 and
 $Q2 = [s/SID : (\exists j/JID)(PROJECT(SJ,j) \wedge (\forall p/PID)(PART(p,A) \rightarrow SUPPLY(s,p,j)))]$.

By applying Rule 3 and 5 to Q, we get

$$IQ1 = \pi_s(IQ1! * s IQ2!).$$

2. Let $Q21 = [j/JID : PROJECT(SJ,j)]$

and $Q22 = [s/SID, j/JID : (\forall p/PID)(PART(p,A) \rightarrow SUPPLY(s,p,j))]$

From Rule 3 and 5, we obtain

$$IQ21 = \pi_j(IQ21! * j IQ22!).$$

3. Let

$R1 = [p/PID : PART(p,A)]$ and

$R2 = [s/SID, j/JID, p/PID : SUPPLY(s,p,j)]$

Since R1 is not an empty set,

$$IQ221 = R2 / R1$$

from Theorem 1.

The snapshots of the above evaluation process is shown in Figure 2, where the summary operation is used to evaluate Q22.

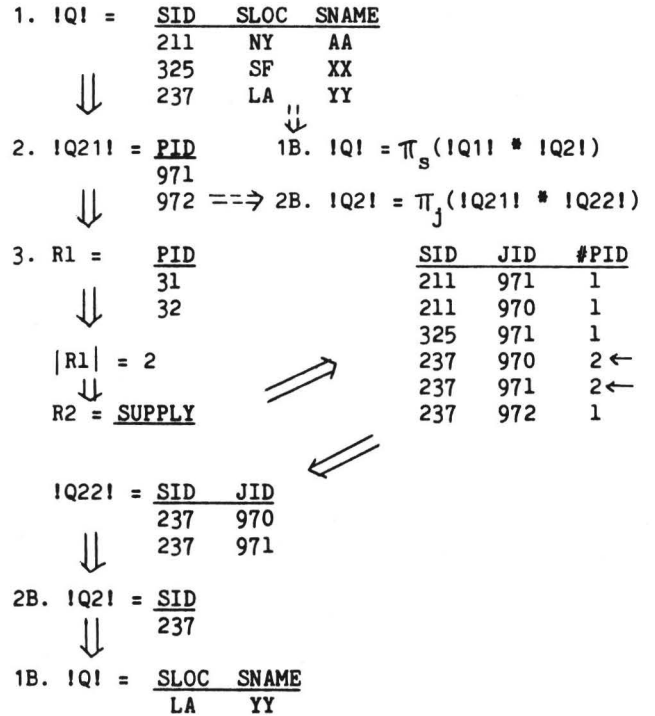


Figure 2. The process of the query evaluation.

4. THE PMVD DECOMPOSITION OF A RELATION

Suppose that the answer of the universally quantified query Q22 in the previous example is saved. Then, we can answer to the same query immediately by simply restoring the saved answer. We need to save answers for other types, e.g. a set of supplier-project pairs (x, y) such that x supplies all parts of type B to y. These answers can be put together if we associate each type to each answers. Let SUPPLYALL (SID, JID, PTYPE) be the relation built in such a way. From SUPPLYALL and PART relations, we can infer the "SUPPLY" fact, namely;

$$(\forall s/SID)(\forall j/JID)(\forall p/PID) ((\exists t/PTYPE) SUPPLYALL(s,t,j) \wedge PART(p,t) \rightarrow SUPPLY(s,p,j)) \quad (4.1)$$

Furthermore, we can remove all tuples which are inferred in (4.1) from the original SUPPLY relation. We rename the reduced SUPPLY relation as SUPPLYSOME. The original SUPPLY relation is now decomposed into two relations: SUPPLYALL and SUPPLYSOME, as shown in Figure 3.

The extension of SUPPLY(s,p,j) can be obtained from not only the SUPPLYSOME relation, but also

SUPPLYALL	SID	PTYPE	JID
	237	A	971
	211	B	970
	237	B	971
	237	A	970

SUPPLYSOME	SID	PID	JID
	211	31	971
	325	32	971
	211	31	972
	237	31	972

Figure 3. The decomposed relations of SUPPLY.

the SUPPLYALL relation. Assume that there are no other relations which contain the information about SUPPLY(s,p,j). Then, the following equation holds:

$$\text{SUPPLY}(s,p,j) = \text{SUPPLYALL}(s,y,j) *_{y} \text{PART}(p,y) \cup \text{SUPPLYSOME}(s,p,j) \quad (4.2)$$

This equation is used to transform queries in terms of simple user's view of relations to those in terms of system's view (Furukawa [1977]). The relational algebraic expression of the query Q22, that is,

$$!Q22! = \text{SUPPLY}(s,p,j) / \varphi_{t=A} \cdot \text{PART}(p,t) \quad (4.3)$$

is transformed to:

$$!Q22! = ((\text{SUPPLYALL}(s,y,j) *_{y} \text{PART}(p,y)) \cup \text{SUPPLYSOME}(s,p,j)) / \varphi_{t=A} \text{PART}(p,t) \quad (4.4)$$

where the operator $\varphi_{f(\vec{t})}$ is defined by

$$\varphi_{f(\vec{t})} R(\vec{x}, \vec{t}) = \{ \vec{x} : R(\vec{x}, \vec{t}) \text{ and } f(\vec{t}) = \text{True} \}$$

and is called the selection with respect to $f(\vec{t})$.

Considering the meaning of the predicate "SUPPLYALL", it is expected that $!Q22! = \{ [s/SID, j/JID : \text{SUPPLYALL}(s,A,j)] \}$ should be included in the answer set $!Q22!$. In fact, we can deduce $!Q22!$ as the answer.

To deduce the result, we need several rules and assertions as follows.

Rules.

RU1. If $R1 = R2 *_{t} R3'(t)$ and

$$R3 = \varphi_{t=ai} R3'(t), \text{ then } R3 \mid R1.$$

RU2. If $R3 \mid R1$, then

$$(R1 \cup R2) / R3 = R1 / R3 \cup R2 / R3.$$

RU3. If x uniquely determines y , then

$$(R1(y,z) *_{y} R2(x,y)) / \varphi_{t=ai} R2(x,t) = \varphi_{y=ai} R1(y,z).$$

Assertions.

AS1. $\text{SUPPLYSOME}(s,p,j) / \varphi_{t=ai} \text{PART}(p,t) = \emptyset$.

AS2. In $\text{PART}(p,t)$, p uniquely determines t .

The deduction process is as follows:

1. From RU1, and (4.4), it is shown that

$$\varphi_{t=A} \text{PART}(p,t) \mid \text{SUPPLYALL}(s,y,j) *_{y} \text{PART}(p,y) \quad (4.5)$$

2. From (4.5), RU2 and (4.4), $!Q22!$ is transformed to

$$(\text{SUPPLYALL}(s,y,j) *_{y} \text{PART}(p,y)) / \varphi_{t=A} \text{PART}(p,t) \cup \text{SUPPLYSOME}(s,p,j) / \varphi_{t=A} \text{PART}(p,t) \quad (4.6)$$

3. From (4.6), AS2, RU3 and AS1, $!Q22!$ is further transformed to:

$$!Q22! = \varphi_{y=A} \text{SUPPLYALL}(s,y,j).$$

This is the final result and is equal to $!Q22!$.

If $\text{SUPPLYSOME}(s,p,j) = \emptyset$, then all suppliers supply only all parts of some types to some projects. In this case, it is said that multi-valued dependency (MVD) $\text{PTYPE} \twoheadrightarrow (\text{SID}, \text{JID})$ holds and it is guaranteed that the SUPPLYPART relation, that is, the hypothetical single relation which includes all information about SUPPLY and PART, can be decomposed into

$\text{SUPPLYALL}(s,t,j) = \pi_p(\text{SUPPLYPART}(s,j,p,t))$ and $\text{PART}(p,t) = \pi_{st}(\text{SUPPLYPART}(s,j,p,t))$ without losing any information.

If $\text{SUPPLYSOME}(s,j,p)$ is not empty, the above condition holds only for the subset (SUPPLY - SUPPLYSOME) of the SUPPLY relation. We say that partial MVD (PMVD) $\text{PTYPE} \twoheadrightarrow (\text{SID}, \text{JID})$ holds in SUPPLYPART relation with SUPPLYSOME relation as exception.

The relation SUPPLYALL, unlike ordinary relations, is a set of intensional data. A tuple (s_0, t_0, j_0) in SUPPLYALL means that the supplier s_0 supplies all parts of type t_0 to the project j_0 . On the other hand, "SUPPLYALL(s_0, t_0, j_0)" may sometimes be interpreted as an extensional fact by itself. (s_0, t_0, j_0) becomes an extensional

fact if it is not unusual that some suppliers supply to some projects all parts of some types. In that case, the query will be expressed directly in terms of "SUPPLYALL", instead of through "SUPPLY".

It has been suggested in Ohsuga [1979] that the information clustering by the division is an important mechanism to build a structure in the data base. The PMVD decomposition schema, together with the associated query transformation equation, enables one to realize the above mechanism in the framework of deductive relational data bases.

We finally remark the evaluation of queries other than universally quantified ones to the decomposed relations. For example, consider the query $Q = [p/PID, j/JID : SUPPLY(237, p, j)]$. By applying the query transformation equation (4.2) to $!Q! = \bigcup_{s=237} SUPPLY(s, p, j)$, it is transformed to

$$!Q! = \bigcup_{s=237} (SUPPLYALL(s, y, j) *t PART(p, y) \cup SUPPLYSOME(s, p, j)). \quad (4.7)$$

We need to perform the costly join of the SUPPLYALL relation and the PART relation to evaluate (4.7). However, it may sometimes be adequate to answer in terms of SUPPLYALL and SUPPLYSOME instead of SUPPLY. In that case, we can avoid the join. It is related to the notion of approximate responses discussed in Joshi et al. [1977].

5. CONCLUSION

Practical algorithms to evaluate universally quantified queries are presented.

Furthermore, a new representation schema based on the PMVD decomposition of relations are introduced. A set of uniform intensional data of the form $(\forall \vec{x}/\vec{c})P(\vec{x})$ is expressed as a relation by using the schema. Universally quantified queries to the decomposed relations are converted to quantifier-free queries by the rule-based symbolic evaluation method.

The DBAP (Furukawa [1977]) are being expanded to realize the algorithms presented in this paper..

ACKNOWLEDGEMENT

The author thank to Dr. Osamu Ishii and Dr. Akio Tojo for providing a chance of the present study, and to Professor Setsuo Ohsuga, Mr. Kazuhiro Fuchi and the staffs of the information systems section for their helpful discussions.

REFERENCES

1. Chang, C. L. [1978] "DEDUCE 2: Further Investigations of Deduction in Relational Data Bases," in Logic and Data Bases, (ed. H. Gallaire and J. Minker), Plenum Press, 1978, 201-236.
2. Codd, E. F. [1972] "Relational Completeness of Data Base Sublanguages," in Data Base Systems, (Ed. R. Rustin), Prentice-Hall, 1972, 65-98.
3. Furukawa, K. [1977] "A Deductive Question Answering System on Relational Data Bases," Proc. Fifth IJCAI, 1977, 59-66.
4. Joshi, A. K. et al. [1977] "Approximate Responses from a Data Base Query System: An Application of Inferencing in Natural Language," Proc. Fifth IJCAI, 1977, 211-212.
5. Kellog, C. et al. [1978] "Deductive Planning and Pathfinding for Relational Data Bases," in Logic and Data Bases, 1978, 179-200.
6. Minker, J. [1978] "An Experimental Relational Data Base Systems on Logic," in Logic and Data Bases, 1978, 107-148.
7. Ohsuga, S. [1979] "Toward Intelligent Interactive Systems," Proc. The IFIP W.G. 5.2 Workshop Seillac II on Methodology of Interaction, (Ed. P. Ten Hagen), North-Holland, 1979.
8. Palermo, F. P. [1974] "A Data Base Search Problem," in Information Systems, (Ed. J. T. Tou), Plenum Press, 1974.
9. Reiter, R. [1977] "An Approach to Deductive Question-Answering," BBN Report No. 3649, Bolt, Beranek and Newman, Inc., 1977.
10. Reiter, R. [1978a] "Deductive Question Answering on Relational Data Bases," in Logic and Data Bases, 1978, 149-178.
11. Reiter, R. [1978b] "On Closed World Data Bases," in Logic and Data Bases, 1978, 55-76.
12. Whitney, V. K. [1974] "A Relational Data Management System," in Information Systems, (Ed. J. T. Tou), Plenum Press, 1974.