<table>
<thead>
<tr>
<th>记录</th>
<th>内容</th>
</tr>
</thead>
<tbody>
<tr>
<td>标题</td>
<td>Relational Strategies for Processing Universally Quantified Queries to Large Databases (Mathematical Methods in Software Science and Engineering)</td>
</tr>
<tr>
<td>作者(s)</td>
<td>FURUKAWA, KOICHI</td>
</tr>
<tr>
<td>引用</td>
<td>数理解析研究所講究録 (1979), 363: 165-187</td>
</tr>
<tr>
<td>发表日期</td>
<td>1979-09</td>
</tr>
<tr>
<td>URL</td>
<td><a href="http://hdl.handle.net/2433/104566">http://hdl.handle.net/2433/104566</a></td>
</tr>
<tr>
<td>类型</td>
<td>Departmental Bulletin Paper</td>
</tr>
<tr>
<td>版本</td>
<td>publisher</td>
</tr>
</tbody>
</table>

京都大学
RELATIONAL STRATEGIES FOR PROCESSING UNIVERSALLY QUANTIFIED QUERIES TO LARGE DATA BASES

Koichi Furukawa
Computer Science Division, Electrotechnical Laboratory
Tokyo, Japan

ABSTRACT
A new algorithm is developed to process universally quantified queries under the closed world assumption. Our algorithm evaluates a universally quantified conditional query very efficiently by reducing the problem to the application of the operations of set intersection, summary and join to the answers to subqueries. Furthermore, the algorithm is extended to handle numerical quantifiers.

The sum-of-product decomposition of a relation is introduced as an efficient representation schema to express a set of uniform intensional data of the form \((\forall \vec{x} / \exists \vec{y}) P(\vec{x})\). Queries to the original relation are transformed to those to the decomposed relations using a query transformation axiom. Furthermore, universally quantified queries to those relations are converted to quantifier-free queries by the symbolic division.

Keywords: deductive question answering, relational database, relational algebra, closed world assumption, query transformation, query evaluation, universal quantifier, numerical quantifier.
1. INTRODUCTION

It has been shown in Chang [1978], Furukawa [1977], Minker [1978], Kellog 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. In his algorithm, universally quantified queries are converted to the application of the division operation to the answers to the subqueries. Parelmo [1974] has improved Codd's algorithm by planning the evaluation process locally to keep intermediate results as small as possible.
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. The above strategy is considered to perform a global planning and optimization.

Furthermore, the algorithm is extended to handle numerical quantifiers such as "at least two", "exactly three" or "more than a half".

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 $(\forall x/\exists y)P(x)$. In this paper, a new representation schema based on the sum-of-product decomposition of a relation is introduced as a suitable way to express such data. It will be shown that the above representation schema increases the
power of expression of relational data bases nearly as high as that of hierarchical data bases.

Queries to the decomposed relations are evaluated using a set of axioms 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 a symbolic division.

In this paper, proofs of the theories are omitted because of the limit of the space.

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 \([x_1/\tau_1, \ldots, x_n/\tau_n: (q_1y_1/\theta_1) \ldots (q_my_m/\theta_m) W(x_1, \ldots, x_n, y_1, \ldots, y_m)]\) where \(W(x_1, \ldots, x_n, y_1, \ldots, y_m)\) is a quantifier free formula with free variables \(x_1, \ldots, x_n, y_1, \ldots, y_m\) and \(q_i\) is either \(\forall\) or \(\exists\), \(i = 1, \ldots, m\). We shall use the abbreviated notation \([\overline{x/\tau: (q\overline{y}/\theta)} W(\overline{x, y})]\) to express a typical query. The \(\tau\)'s and \(\theta\)'s, which are called types, are sets of constant signs which the variables associated with them range over. A sequence of types \(\overline{\tau} = (\tau_1, \ldots, \tau_n)\) is the set \(\prod_{i=1}^n x_i \ldots x_{\tau_n}\).

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 \( \text{!Q}_\text{CWA} \) to \( Q = [\overline{x}/\overline{z} : (Qy/\theta)w(\overline{x}, \overline{z})] \) is defined as follows:

\[
\text{!Q}_\text{CWA} = \{ \overline{c} : \overline{c} \in \mathcal{I} \text{ and } \text{DB} \cup \overline{EDB} \vdash (Qy/\theta)w(\overline{x}, \overline{z}) \}
\]

where \( \overline{EDB} = \{ \overline{P}c : P \text{ is a predicate sign, } \overline{c} \text{ a tuple of constant signs and } \text{DB} \not\vdash P\overline{c} \} \).

It seems to be impractical to get the set of CWA answers to \( Q \) since \( \overline{EDB} \) may contain infinite numbers of literals. But it has been shown in Reiter [1978a] that the CWA answers to an atomic query can be obtained without \( \overline{EDB} \), i.e. \( \text{!Q}_\text{CWA} = \text{!Q}_\text{OWA} \), if \( \text{DB} \cup \overline{EDB} \) is consistent (\( \text{!Q}_\text{OWA} \) is a set of minimal answers to \( Q \) under the open world assumption). Henceforth, we consider only the CWA answers and abbreviate \( \text{!Q}_\text{CWA} \) to \( \text{!Q} \).

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 \( Q = [\overline{x}/\overline{z}, z/y : W] \), and \( \overline{x} \) is the \( n \)-tuple \( x_1, \ldots, x_n \). Then \( \text{!Q} \) is a set of \((n+1)\)-tuples, and the projection of \( \text{!Q} \) with
respect to \( z \), \( \pi_z^\ast Q! \) is the set of \( n \)-tuples obtained from \( Q! \) by deleting the \((n+1)\)st component from each \((n+1)\)-tuple of \( Q! \).

Let \( Q = [\bar{x}/\bar{y}, z/\psi : W] \). Then the quotient of \( Q! \) by \( z \), \( \Delta_z^\ast Q! \), is a set of tuples and is defined as follows:

\[ (\bar{z}, a) \in \Delta_z^\ast Q! \text{ iff } (\bar{z}, a) \in Q! \text{ for all } a \in \psi. \]

The operator \( \Delta_z \) is called the division with respect to \( z \).

Let \( Q_1 = [\bar{x}/\bar{y}, z/\psi : W_1] \) and \( Q_2 = [\bar{y}/\bar{d}, z/\psi : W_2] \). The join of \( Q_1! \) and \( Q_2! \) with respect to \( z \), \( *z(Q_1!, Q_2!) \), is a set of tuples and defined as follows:

\[ (\bar{z}, d, \bar{e}) \in *z(Q_1!, Q_2!) \text{ iff } (\bar{z}, d) \in Q_1! \text{ and } (\bar{e}, d) \in Q_2!. \]

We sometimes denote \( *z(Q_1!, Q_2!) \) as \( (Q_1! *z Q_2!) \).

The set of query conversion rules are listed below:

**Rule 1.** (Decomposition of AND/OR queries)

1. \( ![\bar{x}/\bar{y} : W_1 \land W_2]! = ![\bar{x}/\bar{y} : W_1]! \land ![\bar{x}/\bar{y} : W_2]! \).
2. \( ![\bar{x}/\bar{y} : W_1 \lor W_2]! = ![\bar{x}/\bar{y} : W_1]! \lor ![\bar{x}/\bar{y} : W_2]! \).

**Rule 2.** (Elimination of negation)

1. \( ![\bar{x}/\bar{y} : \overline{W}]! \ = \ \overline{\bar{x}} - ![\bar{x}/\bar{y} : W]! \).
2. \( ![\bar{x}/\bar{y} : W_1 \land W_2]! \ = \ ![\bar{x}/\bar{y} : W_1]! \land ![\bar{x}/\bar{y} : W_2]! \).

**Rule 3.** (Distribution of quantifiers)

1. \( [\bar{x}/\bar{y} : (\exists \bar{x}/\bar{d})(W_1 \lor W_2)] \)
   \[ = [\bar{x}/\bar{y} : (\exists \bar{x}/\bar{d})W_1] \lor [\bar{x}/\bar{y} : (\exists \bar{x}/\bar{d})W_2]. \]
2. \( [\bar{x}/\bar{y} : (\forall \bar{x}/\bar{d})(W_1 \land W_2)] \)
= [x/\bar{\tau} : (\forall y/\bar{\theta})W_1] \land [x/\bar{\tau} : (\exists y/\bar{\theta})W_2].

Rule 4. (Elimination of existential quantifiers)

![x/\bar{\tau} : (\exists y/\bar{\theta})W]! = \pi_y![x/\bar{\tau}, y/\bar{\theta} : W]!

where \pi_y denotes \pi_{y_1}...\pi_{y_m} and \pi_y is the projection with respect to y_i.

Rule 5. (Elimination of universal quantifiers)

![x/\bar{\tau} : (\forall y/\bar{\theta})W]! = \Lambda_y![x/\bar{\tau}, \bar{y}/\bar{\theta} : W]!

where \Lambda_y denotes \Lambda_{y_1}...\Lambda_{y_m} and \Lambda_y is the division with respect to y_i.

Rule 6. (Decomposition of existentially quantified AND queries)

![x/\bar{\tau} : (\exists y/\bar{\theta})(W_1 \land W_2)]!

= (![x_1/\bar{\tau}_1, y/\bar{\theta} : W_1]! \ast ![x_2/\bar{\tau}_2, y/\bar{\theta} : W_2]!)

where x_1 is a sub-sequence of x which appears in W_1, and \bar{\tau}_i is the corresponding sub-sequence of \bar{\tau}, for i = 1, 2. The join operation \ast is performed with respect to the variables y and x_1 \land x_2.

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

1. In rule 2, the restriction that W, W_1 and W_2 must be quantifier free is removed. For example, if W = (\exists y/\bar{\theta})W', then

![x/\bar{\tau} : W]! = ![x/\bar{\tau} : (\exists y/\bar{\theta})W']! = ![x/\bar{\tau} : (\forall y/\bar{\theta})W']!.

Rule 2 claims that

![x/\bar{\tau} : (\forall y/\bar{\theta})W']! = \bar{\tau} - ![x/\bar{\tau} : (\exists y/\bar{\theta})W']!
but not that

$$![\exists/\exists : (\exists \exists/\exists)W']!] = \exists - ![\exists/\exists : (\exists \exists/\exists)W']!,$$

which has been presented as a counter example in Reiter [1978a].

2. Rule 3.2 and Rule 5 are 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 ommitted 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 6 is new, too. It is easily shown that this conversion reduces the amount of computation. Let $$Q_i = [\exists/\exists, \exists/\exists : W_i]$$ and $$\exists^i = \pi_{\exists^i}(\exists),$$ i.e., the compliment subsequence of $$\exists_i$$, i = 1, 2. We would have to evaluate $$Q = [\exists/\exists : (\exists \exists/\exists)(W_1 \land W_2)]$$ according to $$!Q! = \pi_{\exists}(!Q_1! \land !Q_2!),$$ unless we have Rule 6. But it is easily shown that

$$!Q_i! = \exists^i \times ![\exists_i/\exists_i, \exists/\exists : W_i]!$$

for i = 1, 2. By applying Rule 6, 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 $$\Delta \exists!Q!$$ where $$Q = [\exists/\exists, z/\psi : (\exists \exists/\exists)W]$$. The following algorithm generates
the answer $\Delta_2 !Q!$.

**Algorithm D.**

1. Group the answer set $!Q!$ to the query $Q$ by $\vec{x}$ and make a list of $\vec{z}$'s for each $\vec{x}$ (we denote the list by $Z\vec{x}$).
2. For each $Z\vec{x}$, test whether it includes the set $\psi$. If the test succeeds, then put the $\vec{x}$ in the answer set $\Delta_2 !Q!$.

Rule 5 says that $\Delta_2 !Q!$ is the answer set for the universally quantified query $Q_0 = [\vec{x}/\vec{z} : (\forall z/\psi)(q^{\vec{x}/\vec{z}})W]$. Notice that $Q_0$ is the abbreviation of the conditional query $Q_0' = [\vec{x}/\vec{z} : (\forall z/\psi)(z \in \psi \rightarrow (q^{\vec{x}/\vec{z}})W)]$. The inclusion test $\psi \subseteq Z\vec{x}$ in Algorithm D reflects the implication in $Q_0'$, where the consequent $z \in \psi$ in $Q_0'$ corresponds to the lefthand set $\psi$ and the antecedent $(q^{\vec{x}/\vec{z}})W$ to the righthand set $Z\vec{x}$.

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

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

$$!Q_c! = \Delta^{z' \notin !Q_c!} !Q! \quad \text{if} \quad !Q_z! \not\models \phi,$$

$$= \vec{x} \quad \text{otherwise}.$$
\textit{A}. An example of such queries is: "List the departments that sell all items that are supplied"; that is,
\begin{equation*}
[x/\text{DEPT} : (\forall y/\text{ITEM}) \text{ SUPPLIED}(y,x) \rightarrow \text{SELL}(x,y)].
\end{equation*}
We can deal with the general case by introducing the following cut operation:

Let \( R = !\{x/z,w/\psi : W(x,z)\} ! \). Then the \textit{cut of} \( R \) \textit{with respect to} \( x \), denoted by \( f_x(R) \), is a set of pairs \( (c_i, D_i) \) where \( c_i \in z \) and \( D_i \) is a set of \( d_i \in \psi \) such that
\begin{equation*}
D_i = \{d_{i,k} : (c_i, d_i) \in R\}.
\end{equation*}
We regard \( f_x(R) \) to be a function from \( \pi_xR \) to \( 2^{\pi_xR} \) and denote \( D_i \) as \( f_x(R)(c_i) \). \( f_x(R)(c_i) \) is called the \textit{cross-section of} \( f_x(R) \) at \( c_i \).

The set of \( D_i \) is exactly the same as \( Z|c_i \) in Algorithm D. The cut operation corresponds to the group-by operation in SEQUEL2 (Chamberlin et al. [1976]).

\textbf{Theorem 2.} Let \( Q = [x/z : (\forall z/\psi)(W_1 \rightarrow W_2)] \). If \( DB \cup \overline{EDB} \) is consistent, then
\begin{equation*}
!Q! = (z - !(x/z : (\exists z/\psi)W_1)!)
\cup \{c : \phi \models f_x(R_1)(c) \subset f_x(R_2)(c)\},
\end{equation*}
where \( R_i = !(x/z,w/\psi : W_i)! \) for \( i = 1, 2 \).

\textbf{Corollary 2.1.} Let \( Q = [x/z : (\forall z/\psi)(W_1 \rightarrow W_2)] \) where \( W_1 \) does not contain any free variables in \( x \), \( R_1 = !(x/\psi : W_1)! \) and \( R_2 = !(x/z,w/\psi : W_2)! \). If \( DB \cup \overline{EDB} \) is consistent, then
\begin{align*}
!Q! &= \{c : R_1 \subset f_x(R_2)(c)\} & \text{if } R_1 \not\models \phi, \\
&= z & \text{otherwise}.
\end{align*}
Since $A \subseteq B$ 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 $(\tilde{e}_i, |D_i|)$ which summarize the pairs $(\tilde{e}_i, D_i)$.

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.). 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:

$$Q = [x/SNAME, y/SLOC :$$
\((^{3}s/SID)(SUPPLIER(s,y,x))
\land (^{3}j/PID)(PROJECT(SJ,j))
\land (\forall p/PID)(PART(p,A)
\land SUPPLY(s,p,j)))\]

The evaluation of \(Q\) proceeds as follows:

1. Let \(Q_1 = [x/SNAME, y/SLOC, s/SID : SUPPLIER(s,y,x)]\) and 
   \(Q_2 = [s/SID : (^{3}j/JID)(PROJECT(SJ,j) \land (\forall p/PID)(PART(p,A)
   \land SUPPLIER(s,p,j)))]\). By applying Rule 6 to \(Q\), we get
   \(!Q = \pi_{5}(!Q_1 \#_5 !Q_2)\).

2. Let \(Q_{21} = [j/JID : PROJECT(SJ,j)]\) and \(Q_{22} = [s/SID, j/JID : (\forall p/PID)(PART(p,A) \rightarrow SUPPLY(s,p,j))]\). From Rule 6, we obtain
   \(!Q_{2} = \pi_{6}(!Q_{21} \#_6 !Q_{22})\).

3. Let \(R_1 = ![p/PID : PART(p,A)]!\) and 
   \(R_2 = ![s/SID, j/JID, p/PID : SUPPLY(s,p,j)]!\) 
   Since \(R_1\) is not an empty set,
   \(!Q_{22} = \{(s_1, j_1) : R_1 \subset p_{(s,j)}(R_2)(s_1, j_1)\}\)
   from Corollary 2.1.

The snapshots of the above evaluation process is shown in Figure 2, where the summary operation is used to evaluate \(Q_{22}\).

Numerical quantifiers such as "at least two", "exactly
three" or "more than a half" can be neatly dealt with by using the summary operation. We adopt the Chang's notation for numerical quantifiers (Chang [1978]): Let op be one of the relative operators $<$, $\leq$, $>$, $\geq$ or $=$. The fact $(x \text{ op } n)$ is expressed as $(\exists \text{ op } nx)$. For example, $(x > 2)$ is expressed as $(\exists > 2x)$. The following theorems give the algorithms to evaluate numerically quantified queries.

**Theorem 3.** Let $Q = [\mathcal{Z}/\mathcal{Z} : (\exists \text{ op } n)z / \psi]W$ and $R = ![\mathcal{Z}/\mathcal{Z}, z / \psi : W]!$. If $DB \cup \text{ EDB}$ is consistent, then

$$!Q! = \{\mathcal{Z} : |\psi \land p_{\mathcal{Z}}(R)(\mathcal{Z})| \text{ op } n\}.$$

**Theorem 4.** Let $Q = [\mathcal{Z}/\mathcal{Z} : (\exists \text{ op } n)z / \psi](W1 \rightarrow W2)$. If $DB \cup \text{ EDB}$ is consistent, then

$$!Q! = \{\mathcal{Z} : |p_{\mathcal{Z}}(R1)(\mathcal{Z}) \land p_{\mathcal{Z}}(R2)(\mathcal{Z})| \text{ op } n\}$$

where $R_i = ![\mathcal{Z}/\mathcal{Z}, z / \psi : W_i]!$ for $i = 1, 2$.

In order to deal with proportional quantifiers such as "a half" or "80%", we need to slightly expand the Chang's notation. We denote $(x \text{ op } (n/100) \# \text{div})$ as $(\exists \text{ op } n\%x)$ where $\# \text{div}$ is the cardinal number of the divisor $\psi$ in Theorem 3 or the cross-section $p_{\mathcal{Z}}(R1)(\mathcal{Z})$ in Theorem 4. Note that the universal quantifier is equal to the extremal case of the proportional quantifiers, that is, "100%" or $(\exists = 100\%x)$.

4. **THE SUM-OF-PRODUCT 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 \(\text{SUPPLYALL} (\text{SID}, \text{JID}, \text{PTYPE})\) be the relation built in such a way. From \(\text{SUPPLYALL}\) and \(\text{PART}\) relations, we can infer the "SUPPLY" fact, namely:

\[
(\forall s/\text{SID})(\forall j/\text{JID})(\forall p/\text{PID}) \\
((\exists t/\text{PTYPE})\text{SUPPLYALL}(s,t,j) \land \text{PART}(p,t) \rightarrow \text{SUPPLY}(s,p,j)) \quad (4.1)
\]

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

Now, let us consider the query Q22 in the previous example:

\[
Q22 = [s/\text{SID}, j/\text{JID}:(\forall p/\text{PID})(\text{PART}(p,A) \rightarrow \text{SUPPLY}(s,p,j))]. \quad (4.2)
\]

The extension of \(\text{SUPPLY}(s,p,j)\) can be obtained from not only the \(\text{SUPPLYSOME}\) relation, but also the \(\text{SUPPLYALL}\) relation. Assume that there are no other relations which contain the information about \(\text{SUPPLY}(s,p,j)\). Then, the following
equation holds:

\[(\forall s/SID)(\forall p/PID)(\forall j/JID)
\]
\[(SUPPLY(s,p,j)) \equiv
\]
\[(\exists t/PTY)(SUPPLYALL(s,t,j) \land PART(p,t))
\]
\[\lor SUPPLYSOME(s,p,j)). \quad (4.3)\]

This statement is a kind of query transformation axioms which give simple logical views of relations to the users (Furukawa [1977]). It is also considered to be a kind of definitional clauses (Reiter [1977]). The query Q22 is transformed to:

\[Q22 = [s/SID,j/JID : \]
\[(\forall p/PID)(PART(p,A)
\]
\[\land (\exists t/PTY)(SUPPLYALL(s,t,j)
\]
\[\land PART(p,t))
\]
\[\lor SUPPLYSOME(s,p,j))] \quad (4.4)\]

by substituting the righthand expression of (4.3) for the consequent of (4.2). Considering the meaning of the predicate "SUPPLYALL", it is expected that \![Q22! = ![s/SID,j/JID : SUPPLYALL(s,A,j)]! should be included in the answer set \!Q22!. In fact, we can deduce \!Q22! as a part of the answer (actually, \!Q22! is the answer set itself. See Theorem 5). Since the antecedent of Q22 does not include any free variables, we can obtain the answer set \!Q22! by division (Theorem 1). By substituting A for t in the first term of the antecedent of Q22, we get.
SUPPLYALL(s,A,j) \land \text{PART}(p,A). \quad (4.5)

Since SUPPLYALL(s,A,j) and \text{PART}(p,A) does not share any variables, the extension \( R_{spj} \) of the expression (4.5) is equal to the direct product

\[ ![s/SID,j/JID : \text{SUPPLYALL}(s,A,j)]! \times ![p/PID : \text{PART}(p,A)]!. \]

The set \( R_{spj} \) is a part of the dividend of the division. We certainly get ![Q221!] = ![s/SID,j/JID : \text{SUPPLYALL}(s,A,j)]! as a part of the quotient by symbolically dividing the set \( R_{spj} \) by the divisor ![p/PID : \text{PART}(p,A)]!.

We will give a more general description about what we have explained above.

Let \( \text{REL} (\vec{x}, z) \) be an arbitrary relation and \( \text{DIV} (z, t) \) a relation which defines the type \( t \) of each element \( z \) in \( \pi_x (\text{REL}) \). Suppose that there are \( n \) types \( T = \{a_1, \ldots, a_n\} \) of \( z \). We select all \( z \) such that \( (z, a_i) \in \text{DIV} \) and make a unary relation \( \text{DIVai} (z) \) for each \( a_i \) in \( T \). Let \( \text{RELai} (\vec{x}, z) \) be a sub-relation of \( \text{REL} \) which consists of all and only elements \((\vec{x}, d)\) such that \( d \in \text{DIVai} \). Then, it is obvious that

\[ \text{REL} = \bigcup_{a_i \in T} \text{RELai} \quad (4.6) \]

Let \( \text{QUOai} (\vec{x}) \) and \( \text{REMAi} (\vec{x}, z) \) be the quotient and the remainder, respectively, which result from dividing \( \text{RELai} \) by \( \text{DIVai} \); namely,

\[ \text{RELai} = \text{QUOai} \times \text{DIVai} \cup \text{REMAi}, \quad a_i \in T. \quad (4.7) \]
From (4.6) and (4.7), we get

\[ \text{REL} = \bigcup_{a_i \in T} (\text{QUO}_a \times \text{DIV}_i) \cup \text{REM}, \]  
\[ (4.8) \]

where \[ \text{REM} = \bigcup_{a_i \in T} \text{REMA}_i. \]

The relation QUO\_a\_i corresponds to the answer set in the previous example. In order to keep all answers QUO\_a\_i, \( a_i \in T \), in a single relation, we associate the constant type information \( a_i \) to each tuple of QUO\_a\_i and make a new relation QUO\_a\_i' (\( X, T \)) for each \( a_i \in T \). The unified relation QUO (\( X, T \)) is a union of QUO\_a\_i' for all \( a_i \in T \). Since

\[ \pi_t(QUOa'_t \times \text{DIV}) = QUOa \times \text{DIV}_i \]  
\[ (4.9) \]

holds for all \( a_i \in T \), we get

\[ \pi_t(QUO \times t \text{ DIV}) = \bigcup_{a_i \in T} (QUOa \times DIVa). \]  
\[ (4.10) \]

From (4.8) and (4.9), we conclude

\[ \text{REL} = \pi_t(QUO \times t \text{ DIV}) \cup \text{REM}. \]  
\[ (4.11) \]

In (4.11), the relation REL is decomposed into three relations; QUO, DIV, and REM. We shall refer to such a decomposition as a sum-of-product decomposition.

The corresponding logical statement to the equation (4.11) is

\[ (\forall x \neq t)(\forall z \neq t)(REL(x, z) \equiv (\exists t \neq T)(QUO(x, t) \wedge \text{DIV}(z, t)) \cup \text{REM}(x, z)). \]  
\[ (4.12) \]

Some of the universally quantified queries to the decomposed relations are reduced to simpler quantifier-free
queries by applying the following theorem:

Theorem 5. Suppose 1 REL (\(\vec{x}, Z\)) is decomposed to QUO (\(\vec{x}, T\)), DIV (\(Z, T\)) and REM (\(\vec{x}, Z\)) as (4.12), and \[ Q = [\vec{x}/\vec{z} : (\forall z/\forall)(\text{DIV}_{ai}(z) \rightarrow \text{REL}(\vec{x}, z))] \]. If \(DB \cup EDB\) is consistent, then

\[ !Q! = ![\vec{x}/\vec{z} : \text{QUO}(\vec{x}, ai)]! \).

The relation QUO, unlike ordinary relations, is a set of intensional data. It is easily shown from Theorem 5 that \((\vec{c}, ai) \in \text{QUO} \iff (\forall z/Z)(\text{DIV}(z, ai) \rightarrow \text{REL}(\vec{c}, z))\); that is, any tuples in QUO represent corresponding intensional fact. For example, a tuple \((s0,t0,j0)\) in SUPPLYALL means that the supplier \(s0\) supplies all parts of type \(t0\) to the project \(j0\). On the other hand, "QUO(\(\vec{c}, ai\))" may sometimes be interpreted as an extensional fact by itself. \((s0,t0,j0)\) 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 Ohnaga [1979] that the information clustering by the division is an important mechanism to build a structure in the data base. The sum-of-product decomposition schema, together with the associated query transformation axiom, 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 $Q = [p/PID,i/JID : SUPPLY(23,7,p,i)]$. By applying the query transformation axiom (4.3) to $Q$, it is transformed to

$$Q = [p/PID,i/JID : (\exists t/PTY)(SUPPLYALL(23,7,t,i))$$
$$\wedge PART(p,t))$$
$$\lor SUPPYSOME(23,7,p,i)].$$

$!Q!$ is converted to

$$!Q! = ![p/PID,i/JID : (\exists t/PTY)(SUPPLYALL(23,7,t,i))$$
$$\wedge PART(p,t)]!$$
$$\cup ![p/PID,i/JID : SUPPYSOME(23,7,p,i)]! \quad (4.13)$$

by using the set of conversion rules in section 2.

We need to perform the costly join of the SUPPLYALL relation and the PART relation to evaluate the first term in (4.13). However, it may sometimes be adequate to answer in terms of SUPPLYALL and SUPPYSOME 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 as well as numerically quantified ones are presented. They are formulated in the same way so that they can be realized compactly.

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

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

This research is considered to be a step toward a natural language QA system. It has been indicated in Furukawa [1977], Sacedoti [1977] and Harris [1977] that the most difficult problem to be solved to realize such a system is to mesh the user's conceptualization of data with the actual structure of the data. The query transformation mechanism developed here as well as in Furukawa [1977] will help to solve the problem.

**ACKNOWLEDGEMENT**

The author thanks 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**


suppliers

<table>
<thead>
<tr>
<th>SUPPLY</th>
<th>SID</th>
<th>PID</th>
<th>JID</th>
<th>SUPPLIER</th>
<th>SID</th>
<th>SLOC</th>
<th>SNAME</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>211</td>
<td>31</td>
<td>971</td>
<td></td>
<td>211</td>
<td>NY</td>
<td>AA</td>
</tr>
<tr>
<td></td>
<td>325</td>
<td>32</td>
<td>971</td>
<td></td>
<td>325</td>
<td>SF</td>
<td>XX</td>
</tr>
<tr>
<td></td>
<td>211</td>
<td>33</td>
<td>970</td>
<td></td>
<td>237</td>
<td>LA</td>
<td>YY</td>
</tr>
<tr>
<td></td>
<td>211</td>
<td>31</td>
<td>972</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>237</td>
<td>31</td>
<td>972</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>237</td>
<td>31</td>
<td>970</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>237</td>
<td>32</td>
<td>970</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>237</td>
<td>33</td>
<td>970</td>
<td></td>
<td>970</td>
<td>POK</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>237</td>
<td>32</td>
<td>971</td>
<td></td>
<td>971</td>
<td>SJ</td>
<td>X</td>
</tr>
<tr>
<td></td>
<td>237</td>
<td>31</td>
<td>971</td>
<td></td>
<td>972</td>
<td>SJ</td>
<td>Y</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PART</th>
<th>PID</th>
<th>PTYPE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>31</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>A</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>B</td>
</tr>
</tbody>
</table>

Figure 1. An example database.

1. \( !Q! = \pi_S(1Q1 \ast 1Q2) \)  
   \( \Downarrow \)  
   \( !Q! = \pi_J(1Q1 \ast 1Q2) \)

2. \( !Q2! = \pi_P(1Q2) \)  
   \( \Downarrow \)  
   \( !Q2! = \pi_J(1Q21 \ast 1Q22) \)

3. \( R1 = \pi_P(1R1) \)  
   \( \Downarrow \)  
   \( \{R1\} = 2 \)  
   \( \Downarrow \)  
   \( R2 = SUPPLY \)  

\( !Q22! = \pi_J(2Q2) \)  
\( \Downarrow \)  
\( 2B. !Q2! = \pi_S(2Q2) \)  
\( \Downarrow \)  
\( 1B. !Q! = \pi_J(1Q1 \ast 1Q2) \)

Figure 2. The process of the query evaluation.
### SUPPLYALL

<table>
<thead>
<tr>
<th>SID</th>
<th>PTYPE</th>
<th>JID</th>
</tr>
</thead>
<tbody>
<tr>
<td>237</td>
<td>A</td>
<td>971</td>
</tr>
<tr>
<td>211</td>
<td>B</td>
<td>970</td>
</tr>
<tr>
<td>237</td>
<td>B</td>
<td>971</td>
</tr>
<tr>
<td>237</td>
<td>A</td>
<td>970</td>
</tr>
</tbody>
</table>

### SUPPLYSOME

<table>
<thead>
<tr>
<th>SID</th>
<th>PID</th>
<th>JID</th>
</tr>
</thead>
<tbody>
<tr>
<td>211</td>
<td>31</td>
<td>971</td>
</tr>
<tr>
<td>325</td>
<td>32</td>
<td>971</td>
</tr>
<tr>
<td>211</td>
<td>31</td>
<td>972</td>
</tr>
<tr>
<td>237</td>
<td>31</td>
<td>972</td>
</tr>
</tbody>
</table>

Figure 3. The decomposed relations of `SUPPLY`. 