AN ALGEBRA OF QUOTIENT RELATIONS*

Antonio L. Furtado
Departamento de Informatica
Pontificia Universidade Catolica do Rio de Janeiro
Rio de Janeiro, Brazil

Larry Kerschberg
Department of Information Systems Management
University of Maryland
College Park, MD 20742

ABSTRACT

An algebra which operates on partitioned relations is developed. Relation partitioning is achieved by defining equivalence relations on n-ary relations. It is shown that the algebra is as powerful as the original relational algebra, having the advantage of a set-processing capability. This feature provides both greater flexibility in query specification and efficient query processing.

Computing Reviews Classification: 3.70, 4.33, 4.34

Keywords: Relational Model, Relational Algebra, Quotient Relations, Quotient Algebra, Query Language, Intermediate Level Language, High Level Query Language.

INTRODUCTION

The algebra of quotient relations endows Codd's relational algebra [1,2] with explicit set-processing cababilities. The development of this algebra is motivated by several trends in the current literature. They are:

- The inclusion of set-processing capabilities in high level relational languages;
- The emergence of the ANSI/SPARC [3] notion of a conceptual schema to support several external schemata such as the relational, network and hierarchical models; and
- The concern in efficiently evaluating relational operations and expressions.

Most high level relational query languages provide some sort of set-processing capability. QUEL [4], for example, has the SET and AGGREGATE-clauses, while both SEQUEL [5] and SEQUEL 2 [6] have the GROUP BY-clause. Further, SEQUEL 2 has replaced the quantifiers SOME and ALL with a SET operation, a group-qualifying HAVING-clause, and set comparison operators such as IN, CONTAINS, DOES NOT CONTAIN, etc. Query by Example [7] incorporates subsetting operations in two-dimensional query specifications, while SYNGLISH [8] provides special keywords such as SOME, ALL, ONLY, and an EACH-clause to obtain subsetting and aggregation.

The ANSI/SPARC report suggests a data architecture consisting of three schemata: the external, conceptual, and internal schema. If the conceptual schema is to support several differing external schemata, then it must have a data structure and operations general enough to translate among them. Lastly, the efficient evaluation of relational operations and expressions is essential to the overall operating performance of a relational system [9, 10, 11].

The proposed algebra contributes to elucidating the above-mentioned points. The set-processing features of high level languages are incorporated directly into the algebra via the notion of a partition induced by an equivalence on the attributes of an n-ary relation. The blocks of partitioned relations serve as units for relational operators; their representation in other models could provide the flexibility required at the conceptual schema level. Evaluation of relational expressions is efficient since in many cases, it is sufficient to examine a representative of a block to determine whether that block of tuples participates in the resulting relation.

The paper is organized as follows: Section 2 develops the algebra of quotient relations, or quotient algebra, by introducing an equivalence relation on an n-ary relation which partitions it into a collection of disjoint blocks of tuples. Six primitive operations are defined for the algebra, and it is shown to be closed under these operations. Section 3 provides a query specification template, and several examples are given. Universal quantification is replaced by a subsetting criterion allowing logical implication ($p \rightarrow q$) to be stated directly, whereas Codd's relational algebra requires either "division" or a negated version (NOT pvq). Further, relational completeness of the quotient algebra is demonstrated. Several extensions to the algebra appear in section 4, while implementation and efficiency aspects are treated in section 5.

THE ALGEBRA OF QUOTIENT RELATIONS

The definitions and terminology of this section follow that of the relational model [1]. It is

*The authors wish to express their gratitude to the Financiadora de Estudos e Projetos, FINEP, under contract 370/CT and to the National Science Foundation, through grant number MCS 76-03142, for providing partial support for this work.

assumed the reader is familiar with the relational operations; the new algebra is defined in terms of these operations.

(2.1) Definition:

Let $D_1, D_2, D_3, \ldots, D_n$ be sets, not necessarily distinct. A relation R on $D_1, D_2, D_3, \ldots, D_n$ is a subset of the expanded Cartesian product $D_1 \times D_2 \times \ldots \times D_n$; i.e.,

$$R \subseteq D_1 \times D_2 \times D_3 \times \ldots \times D_n$$

The relation R is said to be an n-ary relation or a relation of degree n. Each D_j, $j \in \{1,2,\ldots, n\}$, is called the j^{th} domain of R. An element t of an n-ary relation R is called an n-tuple or simply tuple.

(2.2) Definition:

Let R be an n-ary relation. Corresponding to each D_j, $j \in \{1,2, \ldots, n\}$ is an A_j called an attribute of R which indicates the role of D_j in the relation. The attributes of R are distinct.

(2.3) Definition:

Let I denote the collection of attributes of an n-ary relation R. For a subset A of attributes of I, we denote the complement of A as $\bar{A} = I - A$. The sequence of A is denoted by $\underset{\sim}{A}$, where the attributes are arranged in a fixed sequence. The empty set of attributes is denoted by ϕ.

(2.4) Definition:

Let R be an n-ary relation and A a subset of attributes of R. The relation $\equiv_A \subseteq R \times R$ is said to be an equivalence on R if for tuples t, t' $\in$ R,

$t \equiv_A t'$ iff $t[A_1] = t'[A_1] \wedge \ldots \wedge t[A_k] = t'[A_k]$

where $A_1, A_2, \ldots, A_k \in A$.

Thus if tuples t and t' have the same values with respect to the attributes of A they are equivalent under $\equiv_A$. It is easy to show that $\equiv_A$ is reflexive, symmetric, and transitive, hence an equivalence relation. As such, we say R is partitioned by A (R is considered a set of tuples) into disjoint blocks of equivalent tuples (c.f. [12]), each of which is denoted by

$$[t] = \{ t' | t' \in R \text{ and } t' \equiv_A t \}$$

The collection of all blocks, written as

$$R/A = \{[t] | [t] \subseteq R \text{ and } t \in R\}$$

is called a quotient relation and is represented as a table composed of blocks as depicted in Table 1.

(2.5) Definition:

Let R be an n-ary relation. For different choices of A, different quotient relations are obtained. Together they constitute the family of quotient relations, Q_R, over R;

$$Q_R = \{R/A | A \subseteq I\}$$

(2.6) Proposition: If R is an n-ary relation, then the cardinality of Q_R is 2^n.

This follows directly from the fact that an attribute may or may not participate in a particular A, so there are 2^n possible choices of A, hence one quotient relation induced by each A.

It can be readily verified that for a given n-ary relation R, the family Q_R is a finite lattice with universal bounds the quotient relation $\check{R}$ consisting of a single block, and the quotient relation $\hat{R}$ in which each tuple is itself a block (cf. [12]). The partial order in the lattice is given by the degree of refinement. Thus, for any $R' \in Q_R$ we have $\check{R} \leq R' \leq \hat{R}$ which means that $\check{R}$ is the least refined (consisting of a single block), R' is more refined than $\check{R}$ but less refined than $\hat{R}$ which has a tuple per block.

The algebra to be defined will use quotient relations exclusively. We assume that an n-ary relation R in first normal form [1], which is isomorphic to $\check{R}$, is given initially; other members of Q_R are obtained via the following:

(2.7) Definition:

Let R be an n-ary relation with A and $\mathcal{B}$ denoting any two sets of attributes of R, and R' a member of Q_R. The partitioning operator, / , induces a partition on a quotient relation, while the de-partitioning, * , eliminates a particular partition. The operators must conform to the following rules:

$$(R'/A)/\mathcal{B} \triangleq (R'/\mathcal{B})/A \triangleq (R'/A \cup \mathcal{B})$$

$$(R'/A) * \mathcal{B} \triangleq R'/(A-\mathcal{B})$$

$$\check{R} \triangleq \check{R}/\phi$$

$$\hat{R} \triangleq \check{R}/I$$

In addition we define $\phi - A \triangleq \phi$ and $I \cup A \triangleq I$; it follows, for example, that $R' * \phi = R' = R'/\phi$, $R'/I=\hat{R}$, $R'*I=\check{R}$, and $\hat{R} * A = \check{R}/\bar{A}$.

We now introduce the relations to be used in the examples: R(N,T,O,L), S(N,C,S,T,M), W(N,C,S,T,M), where:

N	employee name
T	team
O	operation
L	location
C	employee's category
S	salary
M	manager name

Table 1 shows $\check{R}$, $\check{S}$ and $\check{W}$ in their tabular representation (which is identical to the relational representation for R, S and W); $Z \leftarrow \check{R}/\{T,L\}$ depicts $\check{R}$ partitioned by T and L.

$\check{R}$ N	T	O	L
b	1	7	x
a	1	3	x
a	1	12	y
a	1	7	x
c	2	3	x
c	2	12	y
d	2	3	x
e	2	3	x

$\check{S}$ N	C	S	T	M
a	1	20	1	b
b	2	30	1	u
c	1	25	2	d
d	2	20	2	u
e	2	20	2	d

$\check{W}$ N	C	S	T	M
f	1	23	1	b
g	1	15	1	b
h	1	20	2	d
i	1	18	2	d
j	1	15	2	d

Z N	T	O	L
b	1	7	x
a	1	3	x
a	1	7	x
a	1	12	y
c	2	3	x
d	2	3	x
e	2	3	x
c	2	12	y

Table 1: Tabular representation of $\check{R}$, $\check{S}$, $\check{W}$, Z

Note that the tuples are grouped into blocks of equivalent tuples. Further, a "flat file" representation is maintained since we are partitioning by sets of attributes rather than sequences of attributes which would imply an order.

The remaining operators of the quotient algebra are projection, restriction, Cartesian product, and union. Although they have counterparts in the conventional algebra, they differ from them in several ways:

- Quotient relational operators must provide appropriately partitioned resultant quotient relations,
- Quotient relational operators deal with blocks of quotient relations as their units (cf. the notion of congruence in [12]).
- θ - comparison operators used in restriction operate on sets of tuples rather than just values.

We now present the remaining operators and discuss how the partitions affect the operators and the evaluation of expressions.

Let R and S be n-ary and m-ary relations, respectively. Consider the quotient relations $R' \in Q_R$ and $S' \in Q_S$ partitioned by attribute sets $\mathcal{A}$ and $\mathcal{B}$, respectively. Further let $\mathcal{C}$ and $\mathcal{D}$ be other attribute sets of R; $\underline{\mathcal{C}}$ and $\underline{\mathcal{D}}$ are sequences of $\mathcal{C}$ and $\mathcal{D}$ (see (2.3)).

(2.8) Definition:

The <u>projection</u>, $T \leftarrow R' [\ \underline{\mathcal{C}}\ ]$, is defined only if $\mathcal{A} \subseteq \mathcal{C}$, in which case T is partitioned by $\mathcal{A}$.

Blocks are treated as units by the operators, and in the case of projection each block is projected on $\underline{\mathcal{C}}$ and possible duplicate tuples are eliminated.

Note that a quotient relation may always be departitioned by I before projecting.

(2.9) Definition:

Let θ denote any one of the <u>set comparison operators</u> in the set $\{=, \underset{\sim}{\sim}, \supseteq, \supset, \subseteq, \subset, \underline{>}, >, \underline{<}, <\}$ which may be modified by an overwritten "/" denoting negation. The last four operators may be modified by "." (to be explained later). The operator "$\underset{\sim}{\sim}$" means that the two sets being compared have at least one element in common, while "$\not\underset{\sim}{\sim}$" means they are disjoint.

(2.10) Definition:

The <u>restriction</u>, $T \leftarrow R' [\ \underline{\mathcal{C}}\ \theta\ \underline{\mathcal{D}}\ ]$, is defined so that a block [t] of R' participates in T only if $[t][\underline{\mathcal{C}}]\ \theta\ [t][\underline{\mathcal{D}}]$, provided that the underlying domains of $\underline{\mathcal{C}}$ and $\underline{\mathcal{D}}$ are θ-comparable.

The resultant quotient relation T is still partitioned by $\mathcal{A}$.

(2.11) Definition:

The <u>Cartesian product</u>, $T \leftarrow R' \otimes S'$, is defined such that for $i \in \{1,2, \ldots, k\}$ and $j \in \{1,2,., p\}$, a block $[t]_{ij}$ of T is defined as

$$[t]_{ij} = [r]_i \times [s]_j,$$

where: x denotes the Cartesian product of the conventional algebra; $[r]_1, [r]_2, \ldots, [r]_k$ denote the blocks of R'; and $[s]_1, [s]_2, \ldots, [s]_p$ denote the blocks of S'.

In the operator of (2.11) the individual blocks of R' and S' are treated as relations to form a block of T. Note that T is partitioned by the disjoint union $\mathcal{A}\ \dot{\cup}\ \mathcal{B}$.

(2.12) Definition:

The <u>union</u>, $T \leftarrow R' \oplus S'$, is defined only if $\mathcal{A} = \mathcal{B}$ and domains corresponding to the attributes in R' and S' are pairwise compatible [2]. Union merges the pairs of blocks from the operands that are equivalent under the common partitioning scheme, and also keeps the unpaired blocks from each operand.

The union $T \leftarrow R' \oplus S'$ is partitioned by $\mathcal{A}$ (or equivalently by $\mathcal{B}$).

QUERY SPECIFICATION IN THE QUOTIENT ALGEBRA

Query specification in the quotient algebra follows a standard format consisting of three well-defined steps. They are:

- The operand relations are partitioned.
- An association between blocks of the operands is specified.
- The associated blocks are then tested by means of a generalized comparison operator.

A number of examples illustrate the use of the algebra to specify and process queries. To avoid ambiguity when attributes with the same name appear in a resulting relation, subscripts will be used. Braces will be omitted in the case of partitioning by singleton sets.

(3.1) Q_1: Find the pairs of teams and locations, such that the team has participated in all operations taking place at a location; together with each team give also all its employees.

Operations	Comments
$A \leftarrow \check{R}[N,T,O]/T$	operations grouped by team
$B \leftarrow \check{R}[O,L]/L$	operations grouped by location
$C \leftarrow A \otimes B$	all pairs team-location are potential candidates
$D \leftarrow C[O_1 \supseteq O_2][N,T,L]$	test if team operations include all operations at the location and project on attributes of interest.

A

N	T	O
b	1	7
a	1	3
a	1	12
a	1	7
c	2	3
c	2	12
d	2	3
e	2	3

B

O	L
7	x
3	x
12	y

C

N	T	O_1	O_2	L
b	1	7	7	x
a	1	3	7	x
a	1	12	7	x
a	1	7	7	x
b	1	7	3	x
a	1	3	3	x
a	1	12	3	x
a	1	7	3	x
b	1	7	12	y
a	1	3	12	y
a	1	12	12	y
a	1	7	12	y
c	2	3	7	x
c	2	12	7	x
d	2	3	7	x
e	2	3	7	x
c	2	3	3	x
c	2	12	3	x
d	2	3	3	x
e	2	3	3	x
c	2	3	12	y
c	2	12	12	y
d	2	3	12	y
e	2	3	12	y

D

N	T	L
b	1	x
a	1	x
b	1	y
a	1	y
c	2	y
d	2	y
e	2	y

Table 2: Processing Q_1

(3.2) Q_2: Find the managers who earn more than their employees.

Servicing this query requires using the "." modifier. In comparing two sets S_1 and S_2 with any of $\{\leq, <, \geq, >\}$, one may want to test the condition for one or all members of either set. For example:

some element of S_1 < all elements of S_2

which is equivalent to testing if (cf. [13]):

$$\min (S_1) < \min (S_2)$$

Herein the modifier "." is used on the side (sides) of the θ-operator where the word all would appear. The various possibilities are shown in Table 3.

$S_1 \backslash S_2$	some	all
some	$<, \leq, >, \geq$	$<., \leq., >., >.$
all	$.<, .\leq, .>, .\geq$	$.<., .\leq., .>., .\geq.$

Table 3: Quantified comparisons

The quotient algebra specification for Q_2 is:

Operations	Comments
$A \leftarrow \hat{S}[N,S]$	each employee as a possible manager
$B \leftarrow \check{S}[S,M]/M$	employees grouped by their managers
$C \leftarrow (A \otimes B)[N=M]$	association of manager-employees
$D \leftarrow C[S_1 >.S_2][N]$	condition concerning salaries

A

N	S
a	20
b	30
c	25
d	20
e	20

B

S	M
20	b
30	u
20	u
25	d
20	d

$\check{D}$

N
b

Table 4: Processing Q_2

(3.3) Q_3: Find the employees in "acceptable" teams, where a team is acceptable if all its present employees (in $\check{S}$) earn at least as much as the prospective ones (in $\check{W}$).

Operations	Comments
$A \leftarrow \check{S}[N,S,T]/T$	present employees grouped by team
$B \leftarrow \check{W}[N,S,T]/T$	prospective employees grouped by team
$C \leftarrow (A \otimes B)[T_1=T_2]$	association by same team
$D \leftarrow C[S_1.\geq.S_2][N_1,T_1] \oplus C[S_1.\geq.S_2][N_2,T_2]$	merge present and prospective employees for teams satisfying the salary condition.

A

N	S	T
a	20	1
b	30	1
c	25	2
d	20	2
e	20	2

B

N	S	T
f	23	1
g	15	1
h	20	2
i	18	2
j	15	2

$\check{D}$

N	T
c	2
d	2
e	2
h	2
i	2
j	2

Table 5: Processing Q_3

The preceding queries were designed to show the ease of query specification in the quotient algebra. Of particular importance is the specification sequence: operand partitioning, association, and condition test, which parallels query specification in the relational calculus [2]. Consider, for example, query Q_3 (ignoring the final merge, for simplicity) as a calculus expression:

$$(s,w):\ \overbrace{s[T] = w[T]\ \wedge}^{\text{association (between blocks)}}$$

$$\forall s' \forall w'\ \underbrace{((s'[T] = s[T] \wedge w'[T] = w[T])}_{\text{partitioning (of operands)}} \rightarrow \underbrace{s'[S] \geq w'[S]}_{\text{condition test}})$$

The quotient algebra allows a more natural query specification than the relational algebra because in the former, logical implication, $p \rightarrow q$, may be expressed directly, while in the latter it must be expressed as NOT p v q. As an example, Q_1 (3.1) would be expressed in the relational algebra as:

Operations	Comments
$A \leftarrow R[T] \times R[O,L]$	all possible combinations
$B \leftarrow A[T,O,L] - R[T,O,L]$	tuples non-existent in $R[T,O,L]$
$C \leftarrow R[T,L] - B[T,L]$	teams having participated in all operations at a location
$D \leftarrow (R[N,T][T=T]C)[N,T,L]$	append employees to accepted pairs

We conclude this section with the

(3.4) Proposition

The quotient algebra is relationally complete in the sense of [2].

Proof: We express each relational algebra operation in terms of an equivalent quotient algebra expression.

relational algebra		algebra of quotient relations
projection:	$R[\underline{A}]$	$\check{R}[\underline{A}]$
restriction:	$R[\underline{A}\ \theta\ \underline{B}]$	$\hat{R}[\underline{A}\ \theta\ \underline{B}] * I$
join:	$R[\underline{A}\ \theta\ \underline{B}]S$	$(\hat{R} \otimes \hat{S})[\underline{A}\ \theta\ \underline{B}] * I_{1,2}$
division:	$R[\underline{A} \div \underline{B}]S$	$((R/\bar{A} \otimes \check{S}[\underline{B}])[\underline{A} \supseteq \underline{B}])[\underline{\bar{A}}] * \bar{A}$
Cartesian product:	$R \times S$	$\check{R} \otimes \check{S}$
union:	$R \cup S$	$\check{R} \oplus \check{S}$
intersection:	$R \cap S$	$((\hat{R} \otimes \hat{S})[\underline{I_1} = \underline{I_2}])[\underline{I_1}] * I_1$
difference:	$R - S$	$((\hat{R} \otimes \check{S})[\underline{I_1} \not\supseteq \underline{I_2}])\ [\underline{I_1}] * I_1$

QUOTIENT ALGEBRA EXTENSIONS

Practical usage may require more than the minimal set of six operators described in section 2. Other operations could be introduced whenever convenient, their definition being given in terms of the basic ones. Obvious choices are the join, intersection and difference operators; some other possible candidates are:

- Conditional expressions, involving more than one comparison and including logical connectives, have been suggested [9, 10] for use with restriction and join.

- Transitive closure is another useful operation. Consider a binary relation $V \subseteq N \times M$, where N is employee name and M is manager name. In order to find the pairs consisting of an employee and his manager's manager, we write in the present algebra:
$$A \leftarrow ((\hat{V} \otimes \hat{V})[M_1 = N_2])[N_1, M_2]$$
which may be merged with $\hat{V}$ to give, for each employee, the two levels of managers to whom he reports:
$$B \leftarrow \hat{V} \oplus A$$
This process may be repeated to m levels of management. Generalizing further, if m is unbounded, additional levels would be added until the result for k + 1 levels contained the same pairs as for k levels.

- Reduction of quotient relations by decomposing them into quotient relations of lesser degree has some theoretical and perhaps future practical interest. The reduction is somewhat analogous to division in the natural numbers, i.e., where a number Z is divided by D to obtain a quotient Q and a remainder R such that Z = QxD+R. Thus quotient relation Z is to be reduced into quotient relations Q and D and some remainder relation R.

Consider $Z \leftarrow \check{R}\ [T,L,O]$ with R as given in table 1.

The computations:

Operations	Comments
$D \leftarrow \check{Z}[L,O]/L$	projection
$Q \leftarrow ((\check{Z}/\{T,L\} \otimes \check{D}/L)[(L_1,O_1) \supseteq (L_2,O_2)])[T,L_1]$	similar to division, with second operand partitioned by non-empty set of attributes.
$R \leftarrow (\check{Z}/\{T,L\} \otimes \check{Q})[(T_1,L_1) \not\sim (T_2,L_2)]$	difference with respect to a specified set of attributes.
$\check{Z} \leftarrow (((Q \otimes D)[L_1{=}L_2])[\bar{L}_2] \oplus R)*I$	reconstruction of $\check{Z}$

yield the results in table 6.

Z

T	L	O
1	x	7
1	x	3
1	y	12
2	x	3
2	y	12

Q

T	L
1	x
1	y
2	y

D

L	O
x	7
x	3
y	12

R

T	L	O
2	x	3

Table 6: Example of reduction

For an n-ary relation Z and attribute sets A, B, such C = A∩B and A∪B = I, the above expressions generalize to:

$$D \leftarrow \check{Z}[\underline{A}]/C$$

$$Q \leftarrow ((\check{Z}/B \otimes \check{D}/C)[\underline{A}_1 \supseteq \underline{A}_2])[\underline{B}]$$

$$R \leftarrow (\check{Z}/B \otimes \check{Q})[\underline{B}_1 \not\sim \underline{B}_2]$$

$$\check{Z} \leftarrow (((Q \otimes D)[\underline{C}_1 = \underline{C}_2])[\bar{\underline{C}}_2] \oplus R) * I$$

Further, whenever $C = \phi$ the formulas for D, Q and for the recomposition of $\check{Z}$ become:

$$D \leftarrow \check{Z}[\underline{A}]$$

$$Q \leftarrow ((\check{Z}/B \otimes \check{D})[\underline{A}_1 \supseteq \underline{A}_2])[\underline{B}]$$

$$\check{Z} \leftarrow ((Q \otimes D) \oplus R) * I$$

so that the formula for Q becomes equivalent (cf. (3.5)) to division in the relational algebra.

IMPLEMENTATION CONSIDERATIONS

Two factors influence efficiency in processing quotient relations: expression evaluation and partition implementation. The first deals with processing a single "big operation" consisting of:

- The partitioning of operand relations. This stage can be done in parallel, with each operand handled separately.

- The association between operand blocks. In a block, tuple values for partitioning attributes are obviously the same. Since the association between blocks depends only on such "representative values", this stage can be performed very efficiently.

- The testing of a θ-condition. Here improved performance may be obtained because not all tuples need be examined in certain cases (e.g. when "$\not\approx$" is used to compare sets, we may "accept" the block as soon as the first match occurs). Also, for order comparisons, the minimum and maximum values of sets can be determined separately for each operand; then only one comparison is needed to test the condition.

Certain efficiency considerations often made concerning the relational algebra are also applicable here. Clearly, one should avoid actually computing a Cartesian product; by looking ahead to the next (simple) operations (especially projections) one may verify if the desired result will in fact be a concatenation of tuples, or a merge, or if one of the operands is simply used to restrict the other (see Q_1, Q_3, and Q_2, respectively, and cf. the "keep" feature in [14]).

Partitions can be implemented in several ways; two are: pointer arrays and aggregate inverted lists. A pointer array P indicating the first tuple in each block of a quotient relation can be used if the tuples have been grouped by some sorting on the partitioning attributes. If physical rearrangement of tuples were undesirable, P could point to another pointer array Q which would indicate tuple rearrangement.

Aggregate inverted lists for a set of attributes A are obtainable, for example, by intersecting the inverted lists for the individual attributes in A (if such inverted lists are available).

A sort routine may be a major tool in an implementation. Sorting on the attributes in A (in any sequence) is one way to perform the partitioning; sorting also on "pivot" attributes may be useful, leading to algorithms linear in the number of comparisons [11]; sorting on attributes in A in a specified sequence can provide a hierarchical presentation of the data for output.

Lastly, the quotient algebra could be implemented quite efficiently on a special purpose processor such as RAP, the Relational Associative Processor

[15]. The "mark bits" associated with each tuple could indicate the block to which it belonged. Relation partitioning would be extremely fast due to RAP's associative processing capability, and its versatile instruction set, powerful enough to support the quotient algebra.

CONCLUSIONS

A relationally complete algebra of quotient relations has been presented. Six fundamental operations comprise the algebra: the partitioning and departitioning operators, together with projection, restriction, Cartesian product, and union.

By endowing the relational model with set-processing capabilities, the quotient algebra incorporates features present in high level query languages. It is hoped that the algebra could serve as an intermediate language into which the high level languages might be translated.

Conversely, the algebra may influence language design. Recent human factors research [16] indicates that certain SEQUEL features involving the GROUP BY-clause, correlation variables and computed variables are difficult to use, even by sophisticated users. The quotient algebra's query format suggests that relation partitioning be performed first, followed by association and condition testing. This fact could be incorporated into a language such as SEQUEL, as shown by a SEQUEL and COBOL-like syntax to represent $Q_1(3.1)$:

```
PARTITION  R[N,T,O]  BY T GIVING A

PARTITION  R[O,L]    BY L GIVING B

ASSOCIATE  A,B  GIVING C

SELECT     N,T,L

FROM       C

WHERE      SET (C[O1]) CONTAINS SET (C[O2]).
```

Partitioned relations and the quotient algebra are possible candidates as data structure and operations for conceptual schema. For example, if R is an n-ary relation with A a collection of attributes of R such that $\bar{A}$ is functionally dependent [1] on A, i.e. $A \rightarrow \bar{A}$, then the blocks of $R' \leftarrow \check{R}/\bar{A}$ can be represented as CODASYL set occurrences; the common element in each block of $R'[\bar{\underline{A}}]$ becomes the owner record, while the elements of $R'[\underline{A}]$ become the respective member records. Functional dependence ensures that an element of $R'[\underline{A}]$ appears in only one block, and therefore cannot have more than one owner as required [17]. A full development of this area is a topic of ongoing research.

ACKNOWLEDGEMENTS

The authors are grateful to M. Stonebraker, C. J. Date, E. Neuhold, and B. Shneiderman for several helpful discussions. Any errors are of course the authors' sole responsibility.

REFERENCES

1. Codd, E. F., "A Relational Model of Data for Large Shared Data Banks", Comm. ACM, Vol. 13, No. 6, June 1970, pp. 377-387.

2. Codd, E. F., "Relational Completeness of Data Base Sublanguages", Data Base Systems ed. by R. Rustin.

3. ANSI/X3/SPARC Study Group on Data Base Management Systems Interim Report, FDT, ACM-SIGMOD, Vol. 7, No. 2, 1975.

4. Stonebraker, M. R. and Wong, E., "INGRES: A Relational Data Base System", Proc. National Computer Conference, AFIPS Press, 1975.

5. Chamberlin, D. And Boyce, R., "SEQUEL: A Structured English Query Language", Proc. ACM-SIGMOD Workshop on Data Description, Access, and Control, May 1974, pp. 249-264.

6. Chamberlin, D. D., et al, "SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control", IBM Research Report RJ 1798, San Jose, June, 1976.

7. Zloof, M., "Query by Example", Proc. AFIPS National Computer Conf., AFIPS Press, Montvale, N.J., 1975, pp. 431-445.

8. Kerschberg, L., Ozkarahan, E. A. and Pacheco, J. E. S., "A Synthetic English Query Language for a Relational Associative Processor", Proc. Second International Conference on Software Engineering, San Francisco, Oct., 1976, pp. 505-519.

9. Pecherer, R. M., "Efficient Evaluation of Expressions in a Relational Algebra:, Proc. ACM Pacific 75 Regional Conf., April, 1975, pp. 44-49.

10. Smith, J. M. and Chang, P. Y., "Optimizing the Performance of a Relational Algebra Database Interface", Comm. ACM, Vol. 18, No. 10, Oct. 1975, pp. 568-579.

11. Furtado, A. L. and Brodie, M. L., "A Data Structure for Fast Relational Algebra Operations", Technical Report 7/76, Catholic University of Rio de Janeiro, 1976.

12. MacLane, S. and Birkhoff, G., Algebra, Macmillan Co., New York, 1967.

13. Boyce, R. F. et al, "Specifying Queries as Relational Expressions: The SQUARE Data Sublanguage", Comm. ACM, Vol. 18, No. 11, Nov. 1975, pp. 621-628.

14. Tsichritzis, D., "LSL: A Link and Selector Language", Proc. ACM-SIGMOD International Conf. on Management of Data, June, 1976, pp. 123-134.

15. Ozkarahan, E. A., Schuster, S. A., and Smith, K. C., "RAP - An Associate Processor for Data Management", Proc. AFIPS Conf., Vol. 44, 1975, pp. 379-387.

16. Reisner, P., "Use of Psychological Experimentation as an Aid to Development of a Query Language", IBM Research Report RJ 1707, San Jose, 1976.

17. Nijssen, G. M., "Set and CODASYL Set or Coset" in Data Base Description (edited by Douque and Nijssen) North Holland/American Elsevier, 1975.