RAQL - AN EVOLUTION IN SAS DATA MANAGEMENT

David Burrage **Michael Gilman**
McGill University School of Computer Science

ABSTRACT

SAS enjoys widespread acceptance as a data management system. Most applications, however, involve the writing of sometimes lengthy and complex SAS programs. These programs are usually specific to the data management problem at hand and do not easily generalize to other data management situations. The writing of ad hoc SAS programs to solve changing, often unforeseen data management problems results in heavy demands on programming staff and computer resources.

The relational approach to the management of data provides a comprehensive and integrated solution to many data management problems. Although SAS currently has elements of a relational system, it most importantly lacks a systematized way to manipulate SAS data sets in a true relational manner.

RAQL is a new relational query language which brings a full relational processing capability to SAS. The higher level language of RAQL permits the relational processing capability to be expressed clearly and very concisely. Information in SAS data sets becomes more accessible. Queries on the data are much more easily formulated. Manipulation of SAS data sets becomes more precise and systematic.

The query facility of RAQL combined with the statistical, editing, report writing and graphics capability of SAS results in a very comprehensive data management package.

INTRODUCTION

Few would dispute the advantages of database systems for the management of data, except perhaps on the grounds of the cost of purchasing and maintaining such systems. There is a proliferation of such database systems today, each having its own merits.

THE RELATIONAL APPROACH: THE DATA MODEL

What distinguishes the database systems on a broad level is the model of data on which they are based. There are three such models currently in common use; hierarchical, network and relational. The uniqueness of the relational model lies in the fact that no system of pointers is used to link the data. Relationships in the data are represented in the same manner as the data itself. The advantage here is that this provides a consistent way to represent information in the database. For example, in the SUPPLIER database below, the fact that a given supplier supplies one or more parts is a relationship in the data represented in the SUP_PART data set. The structure of SUP_PART is identical to the other two data sets in the database.

SUPPLIER DATABASE

SUPPLIER

SP_NAME	CITY
ADAMS	ATHENS
BLAKE	PARIS
CLARK	LONDON
JONES	PARIS
SMITH	LONDON

PART

PT_NAME	PRICE
NUT	0.43
BOLT	0.75
SCREW	0.34
CAM	1.45

SUP_PART

SP_NAME	PT_NAME
ADAMS	BOLT
BLAKE	NUT
BLAKE	SCREW
CLARK	NUT
JONES	CAM
SMITH	SCREW

The data model (the relation) is very easy to conceptualize. This allows the user to better 'see' and understand the data in the database. Because of this, operations on the data are performed more easily with the user having greater confidence in the result.

The model of data that a database system uses also strongly influences the kinds of manipulations of the data that are facilitated. The relational data model allows retrieval, updating, inserting and deleting of data in the database to be accomplished with symmetry and simplicity.

For a more thorough presentation of the advantages of the relational data model see DATE Chapter 3. This is a highly recommended book for an introduction to database systems, particularly relational database systems.

THE RELATIONAL APPROACH: THE OPERATORS

The relational approach requires a set of precisely defined powerful operators to manipulate the relations in the database. The operators are powerful in that they operate on entire relations at a time. The user never writes programs to access the relations observation by observation.

The consistency of always working with whole relations enables the user to have a higher level approach to the manipulation of the data. This expedites results as details of processing are not considered.

To illustrate, three relational operators will be briefly described; SELECT, PROJECT and JOIN.

The effect of SELECT is very similar to using the SAS subsetting IF statement.

T=SELECT SUPPLIER WHERE SP_NAME='SMITH'.

The above will select only those observations from the SUPPLIER relation that have SP_NAME of SMITH. The result is stored in a new relation called T.

T	
SP_NAME	CITY
SMITH	LONDON

The effect of PROJECT is similar to using the KEEP feature of SAS. The major difference is that duplicate observations are eliminated.

T=PROJECT SUPPLIER OVER CITY.

The above keeps only the CITY variable of the SUPPLIER relation and stores it in the new relation T. Notice that duplicate observations have been eliminated.

T
CITY
ATHENS
LONDON
PARIS

The logic of JOIN is similar to the MERGE in SAS except that the JOIN is more general and handles one important situation that the SAS MERGE does not (duplicate values of the BY variable in **both** data sets).

T=JOIN SUPPLIER AND SUP_PART OVER SP_NAME.

The above will join the two relations SUPPLIER and SUP_PART over their common variable SP_NAME.

T		
SP_NAME	CITY	PT_NAME
ADAMS	ATHENS	BOLT
BLAKE	PARIS	NUT
BLAKE	PARIS	SCREW
CLARK	LONDON	NUT
JONES	PARIS	CAM
SMITH	LONDON	SCREW

An indication of the conciseness and power of the relational approach is highlighted by the following simple query.

'Get all suppliers supplying a part costing less than 50 cents'.

T1=SELECT PART WHERE PRICE LT .50.
T2=JOIN T1 AND SUP_PART OVER PT_NAME.
T=PROJECT T2 OVER SP_NAME.

T1	
PT_NAME	PRICE
NUT	0.43
SCREW	0.34

T2		
PT_NAME	PRICE	SP_NAME
NUT	0.43	BLAKE
NUT	0.43	CLARK
SCREW	0.34	BLAKE
SCREW	0.34	SMITH

T
SP_NAME
BLAKE
CLARK
SMITH

The way that the foregoing query was answered is typical of the concise query formulation in the relational approach. With the full set of relational operators at the user's disposal, queries of arbitrary complexity can be decomposed into a short sequence of relational operations. This encourages users to pose otherwise difficult queries and promotes ad hoc query formulation.

THE RELATIONAL APPROACH AND SAS

It is our contention that the relational approach to the management of data is appropriate for the SAS user. The ability to easily manipulate SAS data sets as relations is desirable and has positive consequences on database design and programmer productivity.

Unfortunately, SAS as it stands lacks a major component of the relational approach; namely, the relational operators. It is true that SAS programs can be written to emulate the operators, but this would be cumbersome, prone to error and entirely contrary to a high level approach to data management.

RAQL - A RELATIONAL QUERY LANGUAGE FOR SAS USERS

RAQL is a new relational query language which brings a full relational processing capability to SAS. This means that many of the advantages of the relational approach to the management of data can now benefit SAS users.

RAQL provides the full set of relational operators, namely;

SELECT, PROJECT, JOIN, DIVIDE, UNION, INTERSECTION, MINUS, TIMES.

The operators are expressed in the same high level language presented in the foregoing SUPPLIER database examples.

There are two aspects of RAQL that should prove to be attractive to many SAS users.

RAQL IS A HIGHER LEVEL LANGUAGE THAN SAS

RAQL is a higher level language than SAS in that one RAQL statement is equivalent to many SAS statements. For example, consider the following simple problem from the SUPPLIER database: 'Get all CITYs that have a SUPPLIER'.

The SAS solution: It is not enough to just KEEP the CITY variable in the SUPPLIER relation. As duplicate observations must be eliminated, a prior sort followed by the use of the FIRST. construct is required.

```
PROC SORT DATA=SUPPLIER;
BY CITY;

DATA T;
  SET SUPPLIER;
    BY CITY;
    IF FIRST.CITY;
```

The RAQL solution: The RAQL solution requires just one statement. Sorting and duplicate observation elimination are handled automatically.

```
T=PROJECT SUPPLIER OVER CITY.
```

The foregoing illustrates one simple example where one RAQL statement replaces six SAS statements. Other RAQL operations, such as the JOIN, can replace upwards of thirty SAS statements. When it is considered that a given data management process may require a series of operations, the higher level language aspect of RAQL becomes clearly evident.

RAQL IS EMBEDDED IN SAS

RAQL statements can be introduced at any appropriate point in a SAS program. This means that the relational processing capability of RAQL can be used in conjunction with all other SAS programming features without disturbing the program flow. Consider the following program:

```
(1)     DATA SUPPLIER;
(2)       INPUT SP_NAME $ CITY $;
(3)     CARDS;
(4)     ADAMS   ATHENS
(5)     BLAKE   PARIS
(6)     CLARK   LONDON
(7)     JONES   PARIS
(8)     SMITH   LONDON

(9)     RAQL;
(10)      T=PROJECT SUPPLIER OVER CITY.
(11)    ;;;;
(12)    RAQLEND;

(13)    PROC PRINT DATA=T;
```

Statements (1) to (8) create the SUPPLIER data set (relation). Statement (9) executes a SAS macro named RAQL whose effect is to prepare for the entry of RAQL statements. Statement (10) is a RAQL statement which uses the SUPPLIER relation created in the previous DATA step. Statements (11) and (12) terminate the RAQL environment. Statement (13) is a SAS statement which uses the relation created by the RAQL statement.

The fact that RAQL statements can be embedded in SAS is clear. What should be stressed is that all data sets created by SAS are available to RAQL and vice versa. Absolutely no modification to existing SAS data sets is necessary to take advantage of the relational features of RAQL. This means that a relational database environment can be set up with little or no conversion needed to the relevant SAS data sets.

OTHER RAQL FEATURES - THE MACRO FACILITY

RAQL has other features that facilitate the solution of many data management problems. The most important of these features is the macro facility.

The main merit of RAQL's macro facility is that it allows the definition of an even higher level user interface. That is, users can execute a number of RAQL statements with one macro call. The name of the macro, up to 64 characters long, can be chosen to insulate the user from the details of the relational operations. Thus the macro facility provides a method for the relationally-naive user, for example a casual SAS user, to benefit from the relational processing capability of the RAQL system.

Default parameter declaration, parameter substitution and the conditional execution of RAQL statements are all features of the macro facility which aid ease of use.

THE EFFICIENCY OF RAQL

RAQL statements are translated into SAS source program statements. These SAS statements are then immediately executed to ultimately perform the indicated relational operations. The efficiency of RAQL, therefore, is essentially that of SAS as regards execution time.

It is true that certain specific data management problems can be solved more efficiently in SAS alone. Where this is appropriate, this should be encouraged. However, it should be clear that time spent writing and debugging more efficient programs is very often counterproductive when a general query language is available to generate solutions concisely and clearly.

This is the area where RAQL provides increased efficiency; generating solutions to many data management problems concisely and clearly.

RAQL AS A STAND-ALONE SYSTEM

No knowledge of SAS is necessary to use RAQL. RAQL can be used as a stand-alone relational system for those users either not interested in or not knowledgeable of the details of SAS. An exclusive relational environment can be maintained where contact with SAS is minimized.

A significant aspect that allows RAQL to be used as a stand-alone system is its error detection. RAQL checks for over fifty error conditions and, if found, prints clear natural language diagnostics.

EDUCATIONAL USE

The fact that RAQL can be used as a stand-alone system has significance for educational database environments.

The language of RAQL is based on the syntax used in the popular textbook by DATE Chapter 6. RAQL offers all the facilities described in Date's book, and much more, and is implemented so as to run at reasonable expense on any teaching database. Because of this, RAQL is a valuable teaching tool in database courses which are intended to give students 'hands-on' experience with the variety of the latest database models and facilities.

SAS/RAQL - AN INEXPENSIVE RELATIONAL DATABASE SYSTEM

There is an ever-increasing number of commercially available relational database systems in the computer marketplace. The prospective buyer of such a system is confronted with many considerations which often make the purchase decision difficult. These relational database systems are usually quite costly incurring significant expense in installation, conversion of data, maintenance and training; not to mention the price of the software itself. Additionally, these systems often lack facilities for sophisticated statistical analysis, report writing and graphic display.

It is our belief that SAS/RAQL presents a serious inexpensive alternative to purchasing a new relational database system. Although SAS/RAQL lacks some elements of a true relational database system - such as update concurrency, integrity control, user views - SAS/RAQL will be sufficient for many relational database applications. The extensive facilities of SAS combined with the relational processing capability of RAQL provides a comprehensive data management package that is available at very reasonable cost.

CONCLUSION: RAQL - AN EVOLUTION IN SAS DATA MANAGEMENT

There has been a steady evolution of SAS usage as a data management system. The evolution has been primarily spurred on by the availability of a simple data model (SAS data set) and easy to use programming features. Further evolution has been partly restrained by the absence of a systematic higher level data management method. This has resulted in sometimes inefficient procedures for data organization and retrieval, ultimately adversely affecting programming productivity.

RAQL introduces a systematic high level method to SAS data management through a relational processing capability. As RAQL can be directly embedded in SAS, the relational processing capability can be utilized as part of the natural program flow. Thus many of the advantages of the relational approach are now easily available to SAS users through RAQL.

FOR FURTHER INFORMATION

For further information, please contact Michael Gilman, McGill University, School of Computer Science, Burnside Hall, 805 Sherbrooke St. West, H3A 2K6, Tel: (514) 392-4566.

REFERENCES

CODD, E.F Relational Database: A Practical Foundation For Productivity, Comm. of the ACM, Feb. 82, Vol. 25, No. 2.

DATE, C.J. Introduction to Database Systems, 1977, Second Edition, Addison Wesley.

GILMAN M. & BURRAGE D. RAQL User Manual, Second Edition, 1982, McGill University Computing Centre.

INGRAM, W. Implementing Relational Database Management Techniques in SAS, SUGI 1979, page 294.