System R: A Relational Data Base Management System

Morton M. Astrahan, IBM Research Laboratory, San Jose, California
Donald D. Chamberlin, IBM Research Laboratory, San Jose, California
W. Frank King, IBM Research Laboratory, San Jose, California
Irving L. Traiger, IBM Research Laboratory, San Jose, California

INTRODUCTION

System R is a data base management system which provides a high-level, non-procedural relational data interface. The system provides a high level of data independence by isolating the end user as much as possible from underlying storage structures. The system permits definition of a variety of relational views on common underlying data. Data control features are also provided, including authorization, integrity assertions, triggered transactions, a logging and recovery subsystem, and facilities for maintaining data consistency in a shared-update environment.

The relational model of data was introduced by Codd [1] in 1970 as an approach toward providing solutions to the various outstanding problems of current data base management systems. In particular, Codd addressed the problems of providing a data model or view which is divorced from various implementation considerations (the data independence problem) and also the problem of providing the data base user with a very high-level, non-procedural data sublanguage for accessing data. It should be stressed here that the relational model is a framework or philosophy for finding compatible solutions to these and other problems in data base management; the relational approach is thought to make solutions more elegant and perhaps simpler but the approach by itself does not solve these problems. With this caveat in mind, our first purpose is to briefly describe a related set of data base problems which we are attempting to solve in a coherent way following the relational approach. Our solutions are embodied in an experimental prototype data management system called System R which is currently being designed, implemented, and evaluated at the IBM San Jose Research Laboratory.

We wish to emphasize that System R is a vehicle for research in data base architecture, and is not available as a product. Furthermore, the ideas discussed in this paper should not be considered as having product implications.

To a large extent, the acceptance and value of the relational approach hinges on the demonstration that a system can be built which is operationally complete (can actually be used in a real environment to solve real problems) and has performance at least comparable to today's existing systems. With the present state of systems performance prediction, the only credible demonstration is to actually construct such a system, and to evaluate it in a real environment. The point of this paper, then, is to describe the set of problems which are being studied in the System R framework, to discuss the objectives of the system (which amounts to a description or definition of the term operationally complete), and to describe the architecture of the system, including overall structure, interfaces, and functional design.

The System R project is not the first implementation of the relational approach; however, we know of no other system which is really aimed at an operationally complete capability. Other efforts have demonstrated feasibility in various of the related problem areas. For example, both the IS/1 system [2] and the Phase/0 SEQUEL prototype [3] were single-user systems. No concurrent sharing of data was permitted and hence data control, locking, and recovery issues were greatly simplified. The INGRES project [4] at U.C. Berkeley is also single-user oriented. In addition, each of these projects has an incomplete treatment of views, i.e., of providing various views of data to various users.

The next section describes the overall goals of System R and describes the list of capabilities which we believe to be necessary in an operational environment. The following section describes the architecture of the system, and describes in overview terms its major interfaces and the components which support these interfaces

SYSTEM OBJECTIVES

System R is focused on five main goals:

1. To provide a high level, non-procedural relational data interface.
2. To provide the maximum possible data independence for the basic data objects (base relations).
3. To support derived relational views.
4. To provide facilities for data control consistent with the high level of the data interface.
5. To discover the performance trade-offs inherent in this type of data base capability.

First, each of these goals will be discussed and illustrated.

1. High Level Non-Procedural Relational Data Interface

The trend toward higher level languages has long been evident in the programming

domain. Set-oriented data sublanguages were introduced in 1962 in the CODASYL Information Algebra [5]. Codd's ALPHA language [6] and Relational Algebra [7] raised the level of data sublanguages by letting the user specify the properties of the data required without describing the access path or detailed sequence of operations to be used to obtain the data. This trend toward higher level non-procedural programming [8] is aimed at reducing the number of decisions the programmer must make in order to express his problem/solution, and at making the decisions more relevant to the solution (as opposed to being relevant to the programming of a specific computer). Halstead has examined two programs solving the same problem using his software physics techniques [9], one written in ALPHA and the other in DBTG-COBOL and for this case found that the ALPHA solution required 30 times fewer mental discriminations than the lower level solution. This observation should be directly translatable into increased programmer productivity and ease of maintenance. Thus, human productivity is one strong reason for the goal of supporting a high-level, non-procedural data interface.

The other reason for moving in the direction of non-procedural interfaces is related to the optimization of the execution of the program. If the data base were dedicated to a single application, its structure could be optimized for that application only, and the application could be written in terms of that optimized structure. However, in an integrated data base environment, such local optimization is likely to be inefficient. Hence, the system must itself optimize the execution of each application on a data base whose structure is a compromise among the various applications. The non-procedural, high-level specification better reveals the application intent and hence is easier for the system to use as a basis for optimization.

The available relational languages (ALPHA, Relational Algebra) were very formal and required rather much mathematical sophistication on the part of the user. In particular, the ALPHA language is based on the first order predicate calculus. The relational algebra introduces a collection of aggregrate operators (selection, projection, join, division, etc.) which have relational operands and produce relational results. The need to discover more user-oriented, non-mathematical relational languages became apparent and is currently being pursued by several research groups [11,12].

The principal external interface of System R is called the Relational Data Interface (RDI), and provides relationally complete [7] facilities for data manipulation, data definition, and data control. To support high-level, non-procedural, set-oriented applications, the RDI contains the SEQUEL data sublanguage in its entirety. SEQUEL is documented in [10].

Of course, not all requirements can best be met through a non-procedural approach and for this reason the RDI contains single-tuple-oriented operators (FETCH, INSERT, DELETE, REPLACE, etc.) in addition to the set-oriented capabilities of SEQUEL.

We have designed the RDI to be used in two modes:
(a) Directly by an application program (e.g., a COBOL program) which uses RDI operators to access the data base.
(b) As the target of a translator program (a special case of an application program) which is emulating some other type of user interface.

2. Data Independence

Date [13] has defined data independence as the immunity of applications to change in storage structure and access strategy. Often, however, the notion is associated with the ability of a data base system to provide various logical views of the data base; for example to make visible only selected records of a file, and selected attributes of each record. By view, informally we mean a relational window through which an application can access the data base. The term "window" is used to imply that changes to the data base which affect the view are visible to the application. We wish to distinguish these two notions of data independence. In this subsection we address the only first notion of data independence; the second, which we call the support of derived views, is discussed in the next subsection.

Typically, data management systems permit two levels of data definition. The lower level, or "schema", describes the primitive data objects being managed by the system. In System R, these primitive objects are called base relations. The description of a base relation includes the relation name, attribute names, description of the units of each attribute, the domain of each attribute, the order of the attributes within a relation, the order (if any) of the tuples within a relation, etc. In particular, the definition of a base table does not include any information about physical storage or available physical access paths to the data. However, each base relation has a very direct physical representation, i.e., each tuple of the relation has a stored representation. Data independence implies that the base relation can be supported by a variety of physical structures and access strategies.

Clearly data independence is important if a system is to allow growth and meet the changing requirements of various applications. System R provides a rich set of access structures. Any of these can be used to support a given base relation.

3. Support of Derived Views

The higher level of data independence consists of the ability to define alternative views in terms of the primitive data objects. This notion appears in most

contemporary data management systems and the usefulness of such systems depends in large measure on the capability of the system to support derived views.

The inability to support views which differ from the primitive views often leads to programs which are complex, because they are warped to use views which are not natural but can be supported, and which require extensive maintenance as the system changes over time.

As an example of the usefulness of derived views, consider a data base containing the following two types of records: CATALOG (PARTNO,DESC,PRICE) and SALES (SALENO,PARTNO,QSOLD). The CATALOG file is ordered by part number, and gives the description and price of each part. The SALES file is ordered by sale number, and gives the part number and quantity sold for each sale. Suppose we wish to print out all the SALES records for parts which have a price greater than $1000.

We could write a program to scan through the CATALOG file, finding parts with PRICE> $1000; for each such part, a separate scan could be made through the SALES table to find all the corresponding records. This program would be highly procedural; it would require repeated scanning of the SALES table, and would give the system little opportunity to optimize the query by choosing among alternate access paths.

However, if our system permits the specification of derived views, the user might specify a view consisting of the join of the two files, as follows: SALES-CAT (SALENO,PARTNO, DESC,PRICE,QSOLD). The program could then consist of a single scan through the SALES-CAT view. Besides being easier to write, this program would give the system flexibility to take advantage of new access paths which may become available (such as a PARTNO index on the SALES file) without requiring changes in the program.

A major goal of the System R project is to develop and investigate the technology of derived views. This problem has three distinct aspects, each of which is being studied:

(a) Exactly what set of operations on derived views is supportable? As an example of this issue, imagine a request to delete a tuple from the SALES-CAT view described above. Since this view is a join of two underlying files, it is not obvious what actions should be taken on the files to support the deletion. (Should we delete the SALES record but retain the CATALOG record?) For some kinds of view modification requests, there may be several possible actions which would produce the desired result; for other kinds of requests, there may be no possible supporting action. Codd [18] has described some examples of the latter phenomenon.

(b) How should the view be bound to the available physical structures and access paths? This aspect of the binding problem concerns the optimization of the view and

accesses on the view in terms of available access paths, e.g., indexes, sequential scan, etc.
(c) When should binding be performed? For dynamic view definition, the binding must also be dynamic. In System R, we are investigating various binding-time strategies; dynamic binding will occur for dynamically defined views but for certain often-used or very demanding views, the binding will be done statically with (hopefully) an increase in performance.

4. Data Control Facilities

Data Control includes those aspects of a data base system which control the access to and use of data. We distinguish four types of data control, each of which is being investigated in System R.
(a) Authorization. This form of control is the most common type, being present in almost all current systems. Authorization is the mechanism to permit or deny the creation and manipulation of data structures and views by various users. Any user of System R may potentially be authorized to create new tables and views, and to selectively grant authorizations for his objects to other users. The authorization mechanism of System R is described more fully in [14].
(b) Integrity. Integrity control provides a mechanism for enforcing that the data in the data base obeys certain rules or predicates which have been declared to the system. This form of control is typically not found in current data base systems but is left to protocols imbedded in various application programs. In System R, two main types of control facilities are provided: integrity assertions and triggers. Integrity assertions are expressed in the SEQUEL language as predicates about the data in the data base [15]. The system then guarantees the truth of these predicates. Exactly when the system checks an assertion is a function of both the type of assertion and the transaction boundary which caused the assertion to be checked.

Triggers are actions that are invoked when some triggering condition or action is detected. For example, suppose that the DEPT relation contains an attribute NEMPS which represents the number of employees in the department. To maintain the validity of this value, we can declare triggers to update this field whenever an employee is hired, fired, or transferred.
(c) Consistency. Integrity implies the static correctness of the data base and consistency is concerned with the dynamic correctness. Suppose that one application program is transferring a set of employees from Dept. 48 to Dept. 50, while simultaneously another application program is giving raises to all employees in Dept. 50. The interaction of these programs may have the undesirable result that some but not all of the transferred employees receive the raise. Even worse, if the transferring program encounters a failure and backs out its updates, it may develop

that a raise has been given to someone in Dept. 48.

In current systems the application would contain specific statements (e.g., "LOCK DEPT 50") to avoid these problems. A major goal of System R is to eliminate such defensive coding which is not a part of the problem being solved but is related only to the fact that the solution is running in a certain environment. Since the user cannot know in advance the exact environment in which his application will run (perhaps no other users are currently updating employee records; in this case the lock is not needed), the system must provide the control needed to enforce consistency. The approach being pursued is to require that the user define the boundaries of a transaction, which is a sequence of statements to be executed as an atomic unit. The system then requests whatever resources it needs in the run-time environment to guarantee atomicity. Furthermore, this same atomic unit is used as the unit of integrity, i.e., integrity may be suspended within a transaction but it is guaranteed at the transaction endpoints. If a transaction violates integrity at its endpoint, then the transaction is backed out.

(d) Recovery. The fourth aspect of data control is concerned with preserving the integrity of the data if the system experiences a malfunction or if an application backs up either voluntarily or involuntarily, (e.g., as in the case of deadlock). The recovery capabilities of System R include the usual checkpoint/restart functions as well as the ability to back up an ongoing transaction to user-specified points. These capabilities are examples of functions which are required in order to have an operationally complete capability.

ARCHITECTURE AND SYSTEM STRUCTURE

We will describe the overall architecture of Sytem R from two viewpoints. First, we will describe the system as seen by a single transaction, i.e., a monolithic description. Second, we will investigate its multi-user dimensions. Figure 1 gives a functional view of the system including its major interfaces and components. The RDI, as described previously, is the external interface which can be called directly from a programming language, or used to support various other interfaces. The Relational Storage Interface (RSI) is the access-method-like level which handles the access to single tuples of base relations. This interface and its supporting system (Relational Storage System - RSS) is actually a complete storage subsystem in that it manages devices, space allocation, storage buffers (one level store), transaction consistency and locking, deadlock, backout, transaction recovery and logging. Furthermore, it maintains indexes on selected attributes of base relations.

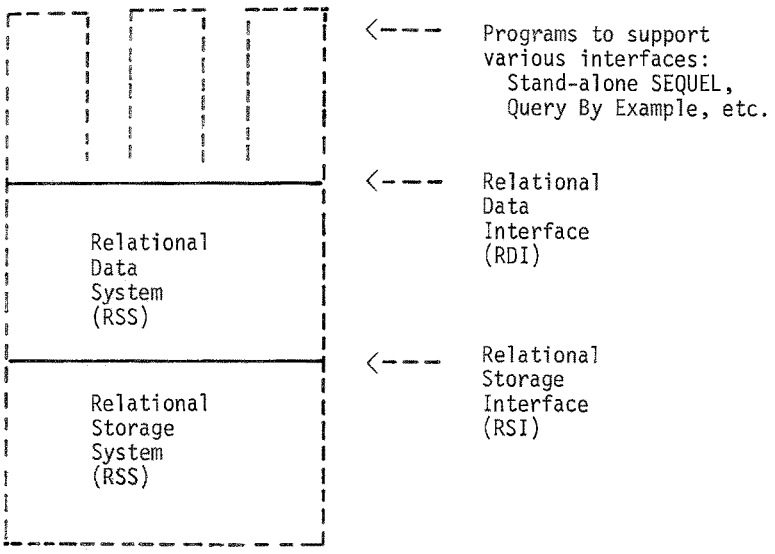

Figure 1

Architecture of System R

With this brief description of the RSS facilities, we can return to the RDI and its supporting system (Relational Data System - RDS). The major functions performed by the RDS are authorization, integrity enforcement, and nonprimitive view support which includes all the binding issues discussed previously. In addition, the RDS maintains the catalogs of external names, since the RSS uses only system-generated internal names. The RDS contains a sophisticated optimizer which chooses the best access path for any given request from among the paths supported by the RSS. The operating system enviornment for this system is VM/370 [16]. Several extensions to this virtual machine capability have been made [17] in order to support the multi-user environment of System R.

ACKNOWLEDGEMENT

The authors wish to acknowledge many helpful discussions with E. F. Codd, originator of the relational model of data, and with L. Y. Liu, manager of the Computer Science Department of the IBM Research Laboratory. We also wish to acknowledge the extensive contributions to System R of Paul L. Fehder, who has transferred to another location, and Raymond F. Boyce, who served as one of the project managers until his untimely death in June of 1974.

REFERENCES

[1] E. F. Codd. A Relational Model of Data for Large Shared Data Banks. Communications of the ACM, June 1970.

[2] M. G. Notley. The Peterlee IS/1 System. IBM UK Scientific Center Report UKSC-0018, March 1972.

[3] M. M. Astrahan and D. D. Chamberlin. Implementation of a Structured English Query Language. Presented at ACM SIGMOD conference, San Jose, California, May 1975; to be published in Communications of the ACM, October 1975.

[4] G. D. Held, M. R. Stonebraker, and E. Wong. INGRES: A Relational Data Base System. Proc. AFIPS National Computer Computer Conference, Anaheim, California, May 1975.

[5] CODASYL Development Committee. An Information Algebra. Communications of the ACM, April 1962.

[6] E. F. Codd. A Data Base Sublanguage Founded on the Relational Calculus. Proc ACM SIGFIDET Workshop, San Diego, California, November 1971.

[7] E. F. Codd. Relational Completeness of Data Base Sublanguages. Courant Computer Science Symposia, Vol. 6: Data Base Systems. Prentice Hall, New York, 1971.

[8] B. M. Leavenworth. Nonprocedural Programming. IBM Research Report RC4968, IBM Research Center, Yorktown Heights, New York., August 1974.

[9] M. H. Halstead. Software Physics Comparison of a Sample Program in DSL Alpha and COBOL. IBM Research Report RJ1460, IBM Research Laboratory, San Jose, California, October 1974.

[10] D. D. Chamberlin and R. F. Boyce, SEQUEL: A Structured English Query Language. Proc. ACM SIGFIDET Workshop, Ann Arbor, Michigan, May 1974.

[11] N. McDonald and M. Stonebraker. CUPID: The Friendly Query Language. Proc. ACM Pacific Conf., San Francisco, California,

April 1975. Available from Boole and Babbage, 850 Stewart Drive, Sunnyvale, California 94086.

[12] M. M. Zloof. Query By Example. Proc. AFIPS National Computer Conference, Anaheim, California, May 1975.

[13] C. J. Date. An Introduction to Data Base Systems. Addison Wesley, 1975.

[14] D. D. Chamberlin, J. N. Gray, and I. L. Traiger. Views, Authorization, and Locking in a Relational Data Base System. Proc. AFIPS National Computer Conference, Anaheim, California, May 1975.

[15] K. P. Eswaran and D. D. Chamberlin. Functional Specifications of a Subsystem for Data Base Integrity. IBM Research Report RJ1601, IBM Research Laboratory, San Jose, California, June 1975.

[16] Introduction to VM/370. IBM Publication No. GC20-1800. IBM, White Plains, New York.

[17] J. N. Gray and V. Watson. A Shared Segment and Inter-process Communication Facility for VM/370. IBM Research Report RJ1579, IBM Research Laboratory, San Jose, California, February 1975.

[18] E. F. Codd. Recent Investigations in Relational Data Base Systems. Proc. IFIPS Congress, Stockholm, Sweden, August 1974.