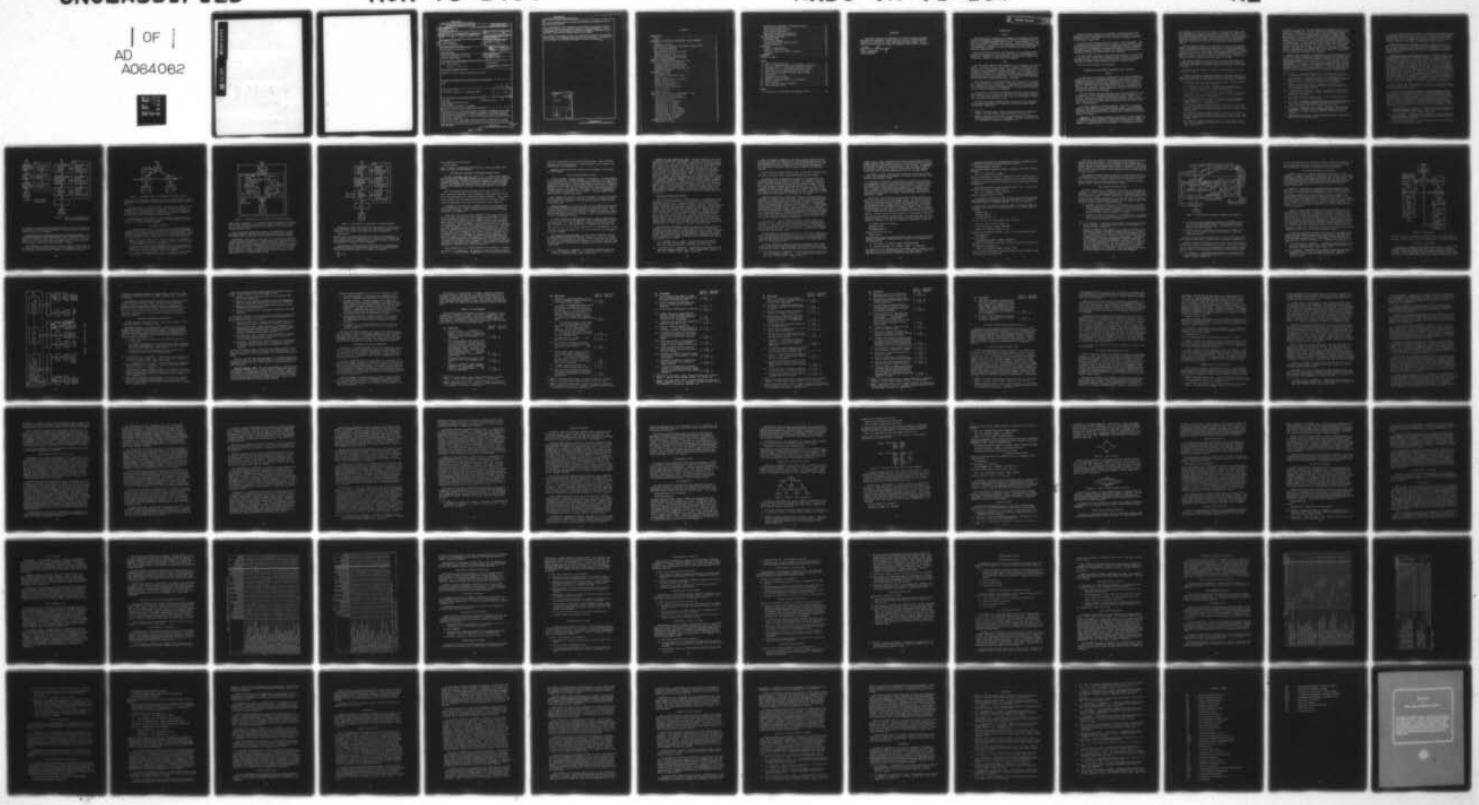
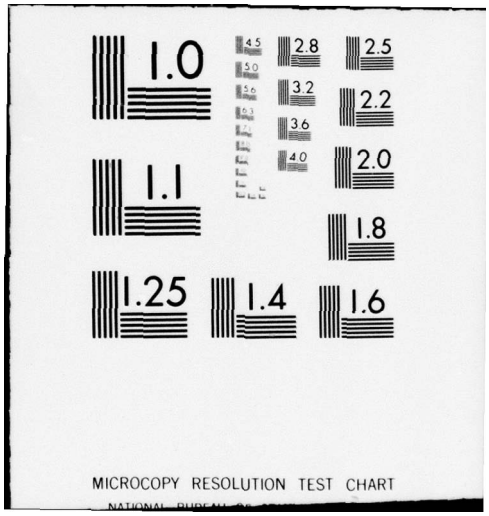
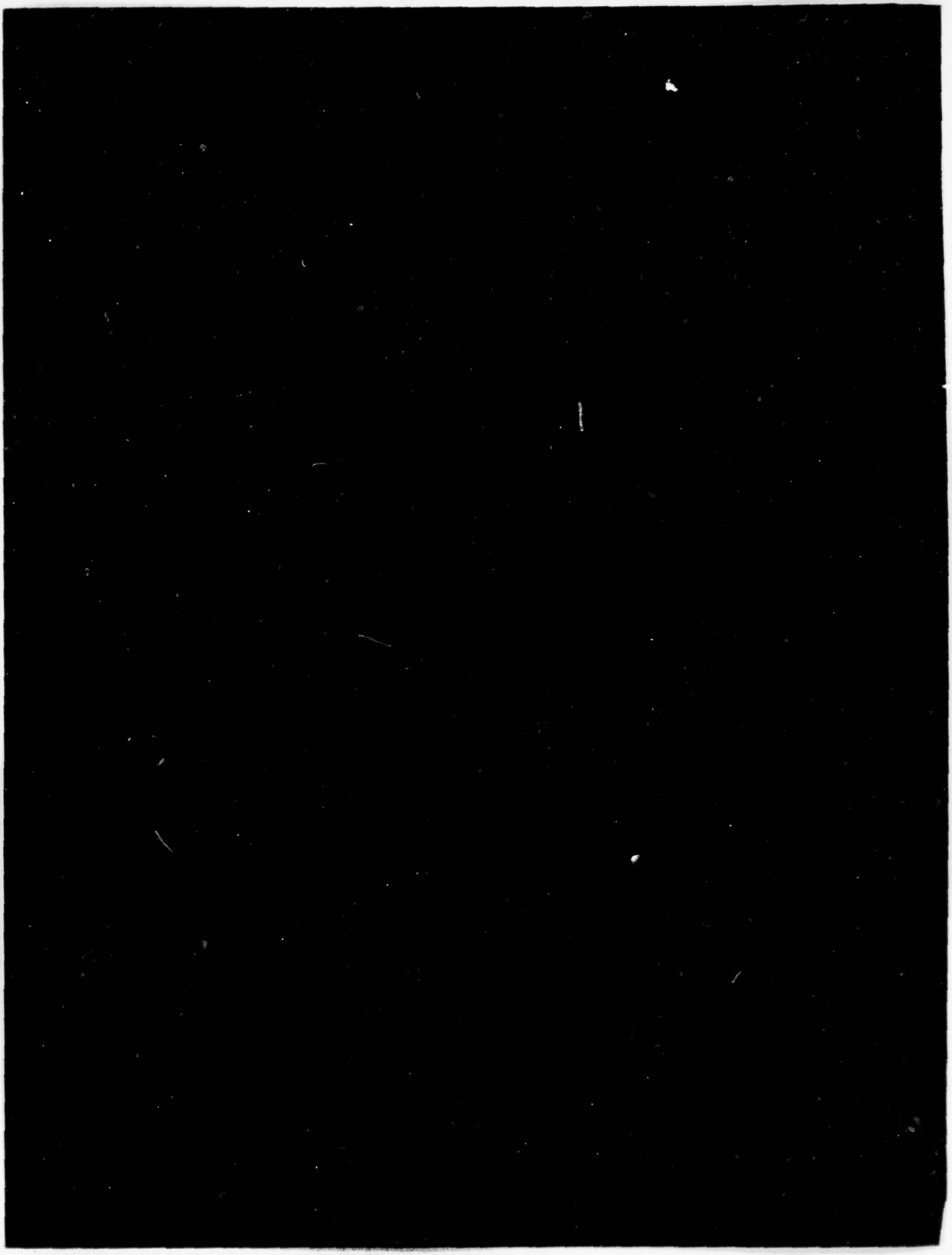
AD-A064 062 MARTIN MARIETTA DATA SYSTEMS DENVER CO F/G 9/2
APPLICATION OF THE REPRESENTATION INDEPENDENT PROGRAMMING SYSTE--ETC(U)
DEC 78 C R SPATH, L S SCHNEIDER F30602-77-C-0142
UNCLASSIFIED MCR-78-1404 RADC-TR-78-197 NL

| OF |
AD
A064062



END
DATE
FILMED
3-79
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: RADC-TR-78-197	2. GOVT ACCESSION NO.
3. RECIPIENT'S CATALOG NUMBER	
4. TITLE (and Subtitle): APPLICATION OF THE REPRESENTATION INDEPENDENT PROGRAMMING SYSTEM TO KNOWLEDGE MANAGEMENT	5. TYPE OF REPORT & PERIOD COVERED: Final Technical Report
6. PERFORMING ORG. REPORT NUMBER: MCR-78-1404	
7. AUTHOR(s): C. Russell Spath, Lowell S. Schneider	8. CONTRACT OR GRANT NUMBER(s): F30602-77-C-0142
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Martin Marietta Corporation, P.O. Box 179, Denver CO 80201	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 62702F, J.O. 45941026, 10
11. CONTROLLING OFFICE NAME AND ADDRESS: Rime Air Development Center (IRDA), Griffiss AFB NY 13441	12. REPORT DATE: December 1978
	13. NUMBER OF PAGES: 78
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office): Same	15. SECURITY CLASS. (of this report): UNCLASSIFIED
82 p.	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE: N/A

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

Same

18. SUPPLEMENTARY NOTES

RADC Project Engineer: P. Langendorf (IRDA)

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Data Base
Distributed Information System
Query System
Representation-Independent Programming
Abstract Data type

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

Brief descriptions of the Representation Independent Programming System (RIPS) components are presented, and preliminary assignment of Knowledge Management (KW) functional requirements to these components are proposed and discussed. The majority of KM functions are found to correspond closely with RIPS concepts and present definitions.
Extensions to RIPS to accomodate some KM functional requirements and current RIPS concepts that extend the KM requirements are discussed and are included

over

DD FORM 1 JAN 73 1473

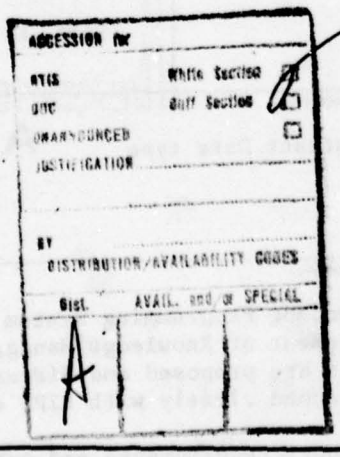
UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

in a final allocation matrix, defining the functional requirements of RIPS to satisfy KM requirements.

The current status of RIPS components are discussed and estimates of time to develop a KM test-bed are provided along with a proposed schedule. Facility requirements for the development are listed and estimated performance characteristics, projected from current results, are provided.

The powerful KM concepts are found to be satisfied by RIPS with few extensions and, in fact, to be enhanced beyond the explicit requirements. Implementation of the concepts in a test-bed environment is found to be feasible.



CONTENTS

Figure

Table

EVALUATION

This report assesses the feasibility of using the currently existing Representation Independent Program System as a means of making known data available as a resource to a very large data base query system. It has direct application wherever large data aggregates are accessed by computer.

Patricia Langendorf

PATRICIA LANGENDORF
Project Engineer

INTRODUCTION

Purpose

The purpose of this document is to establish the feasibility of the Representation-Independent Programming System (RIPS) as an implementation vehicle for a Knowledge Management (KM) test bed. KM requirements[1,2] are analyzed and allocated to existing RIPS components, forming a preliminary functional allocation. Extensions of both KM concepts and RIPS capabilities are then determined, resulting in a final correlation matrix describing RIPS functional specifications.

The current status of RIPS is then discussed, and a development plan is proposed for completion of a prototype system that includes estimates of facilities and manpower as well as projected estimates of performance characteristics. Finally, our conclusions are presented.

Scope

The next section describes RIPS components in the context of an existing system. In reality, RIPS is a collection of concepts, methodologies, and software, still in the research phase. The descriptions are brief, referencing existing documentation and published papers. Some components are in the form of technical notes or papers in progress, and it is impractical to include them. Neither is it practical to develop formal documentation or equivalent details for the current effort. In these cases, only pertinent features and their underlying concepts are described.

The fourth section lists KM requirements taken from Section 2 of Reference 1 and Appendix A.1 of Reference 2. Each requirement is a brief synopsis, sequenced in order of appearance, accompanied by the KM page reference and corresponding references to discussions of its allocation to RIPS components in later sections of this document.

The fifth section contains a preliminary allocation of KM functions to RIPS components. It is organized on the basis of the KM Logical System Design. KM requirements specified in this section are discussed in conjunction with those of the previous section.

The sixth and seventh sections discuss extensions to RIPS required by KM functional requirements, and extensions to KM concepts made possible through RIPS, respectively.

---

1. James F. Berry and Craig M. Cook: *Managing Knowledge as a Corporate Resource*. Contract source document, Version 4.5, 28 May 1976.

2. James F. Berry and Craig M. Cook: *Viewing Knowledge as a Resource in Federal Departments of the U.S. Government*. Economic Research Service, U.S. Department of Agriculture, September 1977.

The eigth section presents the correlation of KM functions with RIPS components, including extensions from the fifth and sixth sections, in an allocation matrix. The result is a final functional assignment, realizing the KM concept through RIPS.

The ninth section discusses the current status of RIPS components. This is an important section because RIPS is a research project and, even though the basic concept appears sound, there is always some risk in the transition from research to development. The technology needed to complete the RIPS functional design to the degree necessary for development is discussed item by item and includes the effect on other RIPS components.

The tenth section contains a work plan for development of a KM test bed or prototype RIPS. Rough order-of-magnitude (ROM) estimates of hardware and software facilities for the test bed are presented, along with estimates of expected performance based on empirical results of current software.

The last section presents our conclusions.

REPRESENTATION-INDEPENDENT PROGRAMMING SYSTEM COMPONENTS

General

The RIPS consists of concepts, methodologies, and software designed to provide solutions to many problems existing in current database systems and Management Information Systems (MISs) implementations and technology. Individual components of RIPS have been developed over a period of several years by the Martin Marietta Database Research Project. Including the results of other researchers--both academic and industrial--incorporated in the RIPS where practical, current progress is the result of more than 100 man-years of research and development.

Each RIPS component has been directed toward a specific problem or class of problems, with the underlying philosophy that mutual compatibility is ensured. There is not a single document describing RIPS. Rather, there are several documents and papers that describe individual concepts and software. Also, some of the more recently developed concepts are still in the form of working papaers.

The section summarizes the RIPS components. References to published papers or other documents are provided when they exist. In the following paragraph, we present a chronology of Database Research Project accomplishments to provide an overview of RIPS evolution and to emphasize the research nature of the project.

Background - The project was formed in early 1974 to perform research on the selection process of Generalized Database Management Systems (GDBMS) and to develop a prototype simulator to evaluate the performance of candidate GDBMSs. An essential element of the simulator is a representation-indepen-

dent statement of the work load or traffic of the application under study. This was satisfied by first stating all data requirements in terms of a canonic representation-independent data model, and second by quantifying all application functions in terms of users and data sources. The canonic model chosen was the Entity Set Model,[3] and quantification of application functions was developed and is referred to as Quantitative Data Description (QDD).[4]

Another essential element of the simulator is a model of the candidate GDBMSs to implement the application under study. The Data-Independent Accessing Model (DIAM I)[3,5] was chosen because of its completeness in describing the various levels of data storage and accessing techniques in a single, standardized model. The DIAM describes the implementation of candidate DBMSs in terms of the Entity Set Model, which was the major reason for selecting the Entity Set Model as the data model for the QDD.

The remaining element of the simulator is a host-computer model describing the performance of candidate computer systems in processing discrete-event application functions generated by the simulator as described by the QDD.

The results of this segment of research and development are in References 6, 7, 8, and 9.

In 1975, the project's scope was expanded to investigate the use of the same concepts employed in developing the simulator for distributed heterogeneous database systems. The objective was to allow users at one node in

---

3. M. E. Senko et al.: "Data Structures and Accessing in Database Systems," *IBM Systems Journal*, No. 1, 1973, pp 30-93.

4. L. S. Schneider and C. R. Spath: "Quantitative Data Description," *Proc. ACM SIGMOD International Conference on Management of Data*, San Jose, California, May 1975, pp 167-195 (ed. W. F. King).

5. M. E. Senko et al.: *A Data-Independent Architecture Model 1: Four Levels of Description from Logical Structures to Physical Search Structures.* IBM Research Report RF 982, February 1972.

6. L. S. Schneider and T. W. Connolly: "Generalized Data Base Management System Simulator," *Proc. 1976 Winter Simulation Conference*, Vol 2, December 1976 (ed. H. J. Highland, et al.).

7. Martin Marietta Database Research Project: *GDMS Math Model Simulator, Functional Specification, Design Specification and User's Guide.* NASA Contract Documentation, NAS9-13951, Johnson Space Center, Houston, Texas, September 1975.

8. Martin Marietta Database Research Project: *GDMS Real-Time Simulator, Functional Specification, Design Specification, and User's Guide.* NASA Contract Documentation, NAS9-13951, Johnson Space Center, Houston, Texas, September 1975.

9. Martin Marietta Database Research Project: *Data Dictionary Research.* NASA Contract Documentation, NAS9-13951, Johnson Space Center, Houston, Texas, September 1975.

a network to access data from other nodes without knowledge of where or how the data are implemented. This work addressed both the end-user facility requirements and mapping of user's queries to the distributed systems. Early in the work, it was recognized that an essential element in this environment is a canonic data model and a sufficiently powerful query language in terms of the data model. Much significant research had already produced useful results in this area using the Relational Data Model [10] as the canonic model. To take advantage of these results, the project chose to replace the Entity Set Model with the Relational Data Model, first demonstrating that DIAM concepts were still valid in terms of the Relational Model.[11] Specifically, this decision was based on the belief that Relational Query Languages such as SEQUEL,[12] QUEL,[13] and others[14,15] would suffice as a representation-independent query language.

However, later investigations determined that existing languages did not offer clear semantic separation of application functions, and despite their promise, the practice of embedding the query lanuage in a general-purpose programming language was still necessary.[12,16] It was not clear that the ANSI-X3-SPARC architecture[17] could be adequately realized because external mappings were still expressed in representation-dependent terms, and access to derived data was limited by the language. To fully implement ANSI-X3-SPARC architecture, it is necessary that user's queries be independent not only of internal representations, but also of external representations.

---

10. E. F. Codd: "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, Vol 13, No. 6, June 1970, pp 377-387.

11. L. S. Schneider: "A Relational View of the Data Independent Accessing Model," *ACM SIGMOD International Conference on Management of Data*, Washington, D.C., June 1976, pp 75-90 (ed. James B. Rothnie).

12. Morton M. Astrahan and Donald D. Chamberlain: *Implementation of a Structured English Query Language*. RJ1464, IBM Research Center, San Jose, California, October 28, 1974.

13. M. Stonebraker, E. Wong, and P. Kreps: "The Design and Implementation of INGRES," *ACM Transactions on Database Systems*, Vol 1, No. 3, September 1976, pp 189-222.

14. M. M. Zloof: "Query by Example," *Proc. National Computer Conference*, AFIPS Press, Vol 44, 1975, pp 431-438.

15. E. F. Codd: "A Database Sublanguage Founded on the Relational Calculus," *Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access, and Control*, San Diego, California, November 1971, pp 35-68.

16. E. Allman, M. Stonebraker, and G. Held: "Embedding a Relational Data Sublanguage in a General-Purpose Programming Language," *ACM SIGPLAN Notices*, Vol II Special Issue, Salt Lake City, Utah, March 1976, pp 25-35.

17. ANSI/X3/SPARC Study Group: *Database Management Systems, Interim Report*. FDT 7, No. 2, ACM, New York, 1975.

Thus, it became necessary to develop a representation-independent programming language, and this was begun in mid 1976. By early 1977, the language concepts were sufficiently defined to complete a conceptual design of the end-user facility[18] and to begin developing prototype software of a sufficient query compiler for distributed heterogeneous database systems.[19]

Before describing RIPS components, we present conceptual views of the ANSI-X3-SPARC architecture and a distributed information system. The purpose of this discussion is to introduce the major components of RIPS as an implementation of ANSI-X3-SPARC architecture in a distributed information system environment.

Conceptual View of ANSI-X3-SPARC Architecture - The ANSI-X3-SPARC study group's proposed architecture[17] presents a view of information systems that is symmetric with respect to the internal and external mappings that occur. The conceptual schema provides a canonic model of the data to which users address their queries and for which implementers provide efficient access. In terms of modules in the architecture, end user functions or external mappings are performed by the End-User Facility (EUF), and internal mappings are performed by the Data Management System (DMS). To perform the mappings, the modules require access to the external and internal schemata, which are user-specified to the extent that flexibility is accommodated by the EUF and DMS, respectively. Thus, user's queries, submitted via the interfacing techniques provided, must be mapped to representation-independent queries in terms of the canonic data model, and subsequently mapped to the database by the DMS. The results must then be presented by the DMS in terms of the canonic data model, then mapped to the user's display device by the EUF.

The architecture is shown in Figure 1 in terms of the EUF, the canonic data model, and the DMS. The role of the data dictionary/directory is shown as the repository of the metadata.

Conceptual View of RIPS as an Example of ANSI-X3-SPARC Architecture - RIPS conforms to ANSI-X3-SPARC architecture, providing modules to perform the mappings. The conceptual view (Fig. 2) shows the corresponding RIPS modules for ANSI-X3-SPARC components. The figure also points out that RIPS does not replace an existing system but rather provides an interface to whatever system is implemented. Thus, the DMS shown in Figure 2 is responsible for accessing stored and derived data under the direction of RIPS. The purpose of this approach is of course that RIPS is intended to provide

---

18. C. R. Spath and L. S. Schneider: "A Generalized End-User Facility for Relational Database Systems," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

19. L. S. Schneider: "A Relational Query Compiler for Distributed Heterogeneous Databases," Submitted for publication in *ACM Transactions on Database Systems*, January 1977.

17. ANSI/X3/SPARC Study Group: *Database Management Systems, Interim Report.* FDT 7, No. 2, ACM, New York, 1975.

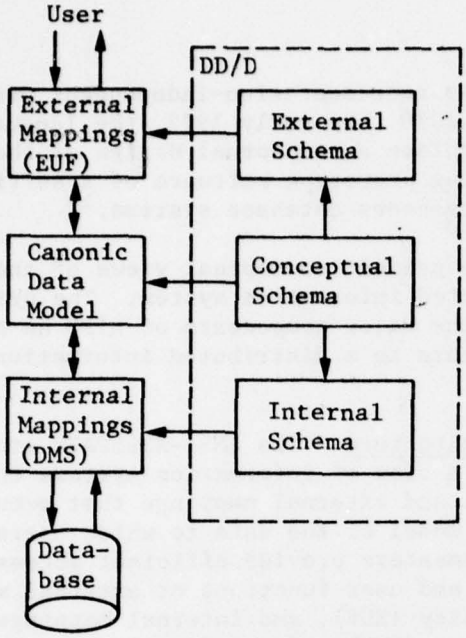


Figure 1. ANSI-X3-SPARC architecture.

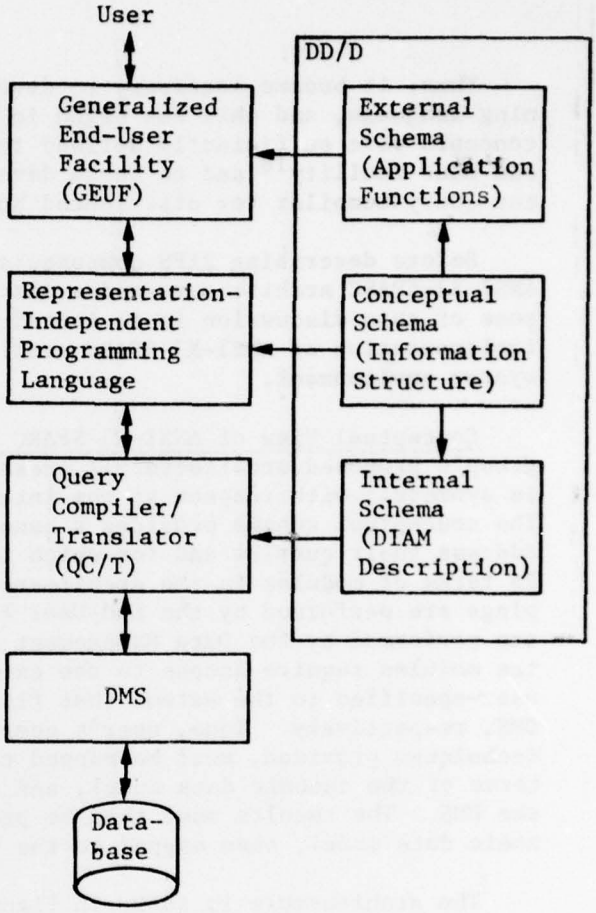


Figure 2. RIPS as an implementation of ANSI-X3-SPARC architecture.

an interface to a distributed heterogeneous database without requiring that the system in the network be reprogrammed to accommodate remote accesses to either its data or its processes.

Conceptual View of a Distributed Information System Environment - In the distributed information system environment, several systems are linked together in a network so that a user at one node may access data, or use processes, at one or more nodes without knowledge of where or how in the network the data are stored or derived, as shown in Figure 3.

The environment we envision is that large and diverse systems exist and are justified, but additional demands arise that suddenly require access to their resources by other systems. Therefore, centralization is not an

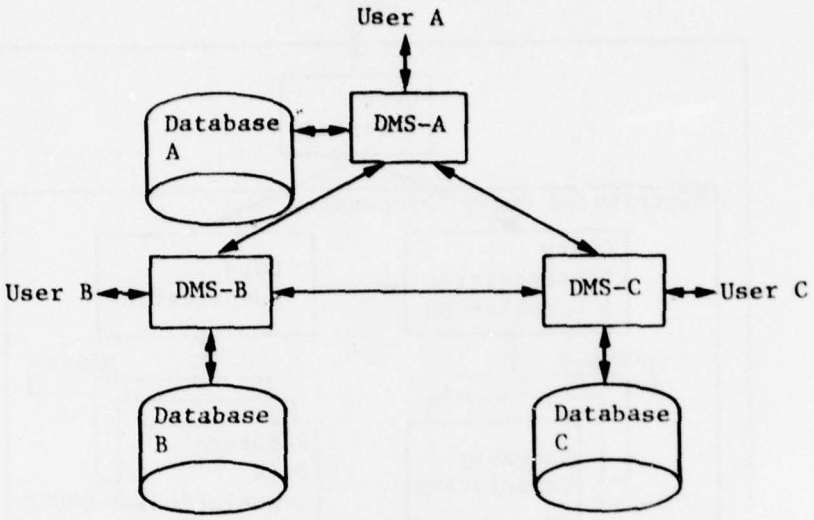


Figure 3. Conceptual view of a distributed information system.

alternative, and, because the systems are large and well established, standardization and transformation are precluded for the reason stated in Reference 2.*

This leads to the view shown in Figure 4 in which user's queries are decomposed into subqueries pertinent to each node, and the subqueries are translated into the language of the target DMS. Of course, a similar process must take place with respect to returned data, either for display or substitution in other subqueries as qualifier values.

Incorporating RIPS architecture in a distributed information system results in a conceptual view shown in Figure 5. We will return to this discussion after describing RIPS components.

Information Structure

The Information Structure (IS) provides the implementation-independent conceptual model in RIPS. The IS is founded on the Relational Data Model,[10] extended to also allow relational descriptions of stored algorithms. At the information structure level, there is no distinction between totally stored data and totally derived data (algorithms), nor is there any distinction

2. James F. Berry and Craig M. Cook: *Viewing Knowledge as a Resource in Federal Departments of the U.S. Government*. Economic Research Service, U.S. Department of Agriculture, September 1977.

* Of course, there may be other environments in which centralization is called for, or in which standardization and/or transformation are justified (e.g., small databases and stable requirements).

10. E. F. Codd: "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, Vol 13, No. 6, June 1970, pp 377-387.

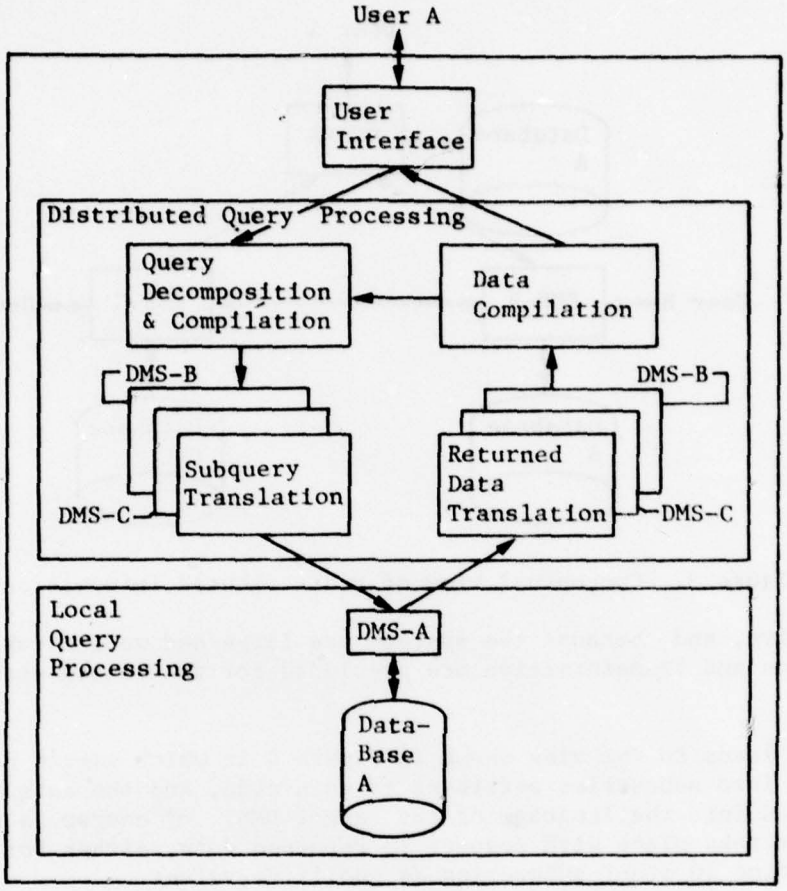


Figure 4. Functional view of a distributed information system.

between data implemented in an automated database and manually stored data. Thus, queries expressed in terms of the IS require no knowledge of where or how the data are implemented, and therefore remain stable in the face of changing implementations.

In a distributed heterogeneous database system environment, a single information structure describes what data are available in the network. This single model is sufficient for both external and internal mappings.

Stored Data - Stored data are described as third-normal-form (TNF) relations. The primary identifier--a single attribute or multiple attributes concatenated--are recorded as such, and secondary identifiers may be declared. The distinction between domain and attribute-role-name is maintained, and both are recorded. A single relation may be declared even though some instances are stored at one node and some at another (i.e., restriction distribution) or some attributes of the relation are stored at different nodes (i.e., projection distribution).

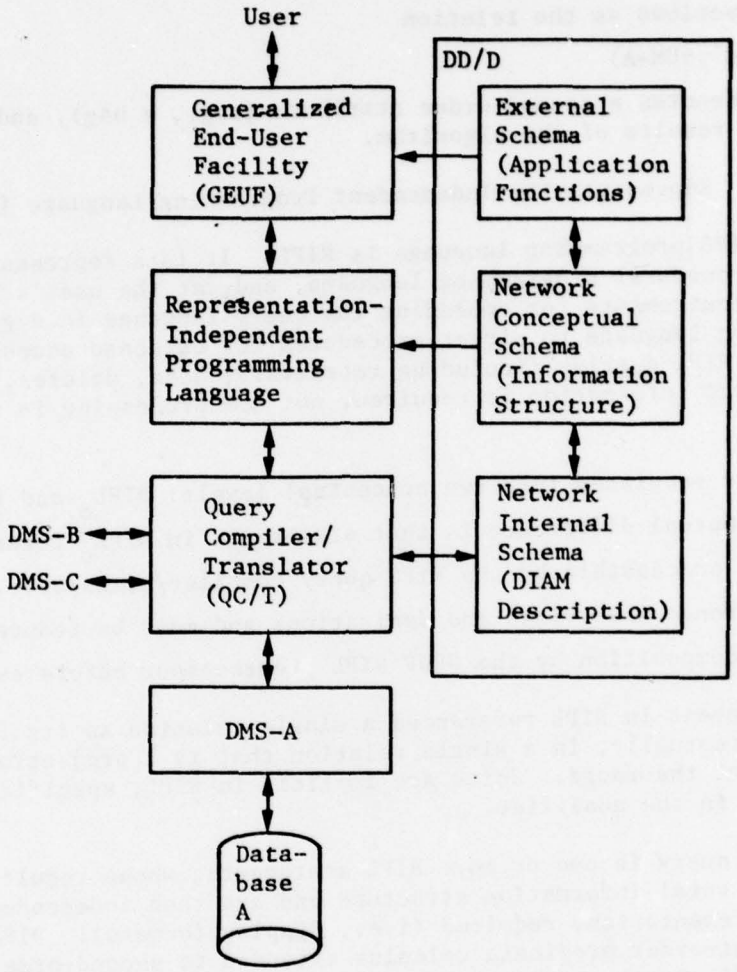


Figure 5. RIPS applied to distributed information system.

Derived Data - Computational functions are described as TNF relations, just as stored data are. For example, the SINE function, computed algorithmically (e.g., Taylor series expansion) can be described as

SINE(θ,SIN-θ)

just as a table of stored values would be. There is no distinction at the IS level. Data derived partially from stored data and partially from stored algorithms (e.g., interpolation over stored values) are described via derivations, discussed in later paragraphs.

An extension of the Relational Data Model allows the representation-independent description of aggregation algorithms by permitting specification of second-order domains. For example, the algorithm

$$\sum_{n=1}^{n} a_f \varepsilon A$$

can be described as the relation

SUM (A',SUM-A)

where A' denotes a second-order attribute (e.g., a bag), and SUM-A represents the results of the algorithm.

Representation-Independent Programming Language (RIPL)

The RIPS programming language is RIPL. It is a representation-independent nonprocedural programming language, and, at the user's level, eliminates current requirements for embedding the query language in a general-purpose programming language to direct procedures for database accesses and computations. RIPL queries (including retrievals, adds, deletes, and changes) specify *what* information is required, not *how* processing is to be accomplished.

RIPL is separated into two conceptual levels: $RIPL_o$ and $RIPL_n$. The major conceptual difference is that statements in $RIPL_o$ reference only the IS and are processable by the RIPS query compiler/translator. $RIPL_n$ statements reference user views and derivations and must be reduced to $RIPL_o$ through decomposition by the GEUF RIPL preprocessor before execution.

A statement in RIPL references a single relation as its range and results, conceptually, in a single relation that is a projection and/or restriction of the range. Joins are implicit in RIPL, specified by linking predicates in the qualifier.

A RIPL query is one or more RIPL statements, whose results form a subset of the total information structure and are thus independent of the external representations required (i.e., display formats). RIPL is founded on the first-order predicate calculus extended to second-order prediction, made possible by the RIPS extended IS. RIPL contains no built-in computational operators, allowing whatever computations are desired to be expressed in a consistent manner. This feature permits RIPL queries to be translated into other languages whose set of built-in operators may differ. For example, if the SUM relation described earlier is referenced in a RIPL query, and the target query language does not contain a built-in SUM operator, the translation must be to a language that does provide the algorithm. Thus, the result of the translation might be a general-purpose programming language that embeds the target query language so that applicable retrievals are in the DBMS language and the summing algorithm is a subroutine call or a procedure in the general-purpose language. Thus, computational operations available for RIPL queries are extensible to include whatever computational algorithms are implemented for a given installation.

$RIPL_o$ contains only two relational operators for prediction: ε (element of) and $\notin$ (not an element of). Any additional relational operators desired (e.g., >, =, ≤, SELLS, EXPORTS, HIGHER-THAN, etc) are declared as derivations

(page 18), and thereafter may be used in $RIPL_n$ queries. Thus, relational operators in RIPL are extensible to include whatever form of prediction is user-defined for an installation.

Examples of RIPL queries and concepts are in Reference 18 and throughout this document.

Data Dictionary/Directory (DD/D)

In RIPS, the DD/D is considered part of the database. Its major purpose is to record metadata. However, metadata are as important to an organization as any other data, and are therefore made accessible with all the same user techniques and facilities. This is accomplished in RIPS by first viewing all metadata as stored relations, and second, including descriptions of these relations in the IS. Thus, queries for metadata are simply representation-independent queries to (a segment of) the database that can be implemented in whatever manner is appropriate for its traffic. The query compiler/translator directs the queries accordingly.

This concept also allows all the same techniques of integrity and authorization management provided for any data to be applied to metadata because these controls are implemented at the information structure level in RIPS, as described later.

Metadata Management - The data directory contains implementation descriptions of both internal storage and external representations of user interfaces and display formats in DIAM descriptions viewed as relations. The directory is available to the query compiler to compile RIPL queries in representation-dependent queries and to the DBA to record and interrogate the details in the management process.

The data dictionary contains metadata that are primarily specifications of user-oriented functions, including user views, predefined queries, integrity assertions, etc. The dictionary is available to the GEUF to provide user interfaces; to the RIPL preprocessor to decompose $RIPL_n$ queries into $RIPL_o$ queries; and to the enterprise administrator in the management of data resources, through the declarations of access controls and the interrogation of current metadata.

Data Description Language (DDL) - The RIPS concept eliminates the distinction between a separate DDL and DML. Because the contents of the DD/D are viewed as relations and are recorded in the IS, queries to add, change, or delete the corresponding metadata instances are simple RIPL queries. Any desired simplified, interfacing user language can be defined using the GEUF capabilities.

---

18. C. R. Spath and L. S. Schneider: "A Generalized End-User Facility for Relational Database Systems," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

Quantitative Data Descriptions (QDD) - The QDD[4] extends the view of the real world, represented by the database, to include representations of the organization itself, and to describe the organization's use of the database. Thus, users (e.g., data sources, product users, etc) are represented in the QDD, along with descriptions of what data they use, how, and at what rate. The composite use--across all users--describes the dynamics of the real world including the organizational use. These constitute the requirements of the system; as the user's profiles change, the system is required to continue to meet the changing demands. User's profiles will change as either the external real world changes or as the organization itself changes. Thus, the database system's implementation is constantly subject to change to accommodate changing requirements. The QDD can be maintained dynamically or recomputed periodically. Either choice is available through the GEUF. The QDD is stored in the data dictionary and is available to the enterprise administrator to provide visibility of organizational data flows or to the DBA to access current requirements. In addition, the QDD is used by the query compiler's optimizer to assess the effects of data populations and population distributions on candidate search paths in the search-path selection process.

Application Function Descriptions -

Partially Predefined Queries - Typical queries to an information system consitute a continuum ranging from ad hoc queries to real-time displays as described in Reference 18. Queries are stated in RIPL in terms of user views, derivations, and/or the basic information structure. Queries that are executed repeatedly are predefined and stored as relations in the data dictionary, and the correlation to the stimulus that will initiate their execution is also declared and stored in the dictionary. Queries that are fully determined as to context and whose stimulus in internally generated (e.g., clock time, other queries, etc) are fully predefined and stored. Queries that are partially predefined, requiring either an externally generated stimulus (e.g., function key, command, light pen, etc) or particularized values of qualification predicates, selected attributes for display, current method of display, etc, are also stored as relations for which the missing values are to be supplied by the user at execution time.

Because partially predefined queries are viewed as relations, the user-supplied values are, in essence, change queries to update the relations. The external represenation of the user-supplied data (i.e., form) is defined by DIAM descriptions stored in the data directory. When the form is transmitted from an external device, The GEUF directs the mapping to the relations to complete the partially predefined query as described under Generalized End-User Facility (GEUF).

---

4. L. S. Schneider and C. R. Spath: "Quantitative Data Description," *Proc. ACM SIGMOD International Conference on Management of Data*, San Jose, California, May 1975, pp 167-195 (ed. W. F. King).

18. C. R. Spath and L. S. Schneider: "A Generalized End-User Facility for Relational Database Systems," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

Thus, the essential elements for partially predefined queries are the relations containing the predefined part, correlation between the query and the applicable form (i.e., stimulus), correlation between the query and the form's description, and the description of the external representation (i.e., geometry of the form). Each of these is specified in relations of the DD/D.

Ad hoc queries are simply stated in $RIPL_n$, and the GEUF decomposes them into $RIPL_o$ as described under Generalized End-User Facility (GEUF).

Any popular method of man-machine interaction is accommodated by the above descriptions and processable by the GEUF. For example, a function key may be defined as the stimulus to execute a predefined query that displays a form; the form may be declared as the stimulus to execute one or more other queries--returning either data, another form, or both, creating a dialogue or computer-aided instruction (CAI). Because the query is independent of both the form's geometry and the stimulus, forms may be changed with respect to the external representation; and other stimuli may be changed without affecting the query or the display of its results.

Stimulus Specifications - Stimuli that initiate predefined queries are specified in relations stored in the DD/D. Stimuli may be either externally supplied (e.g., forms, function keys, light pen, external event monitors, etc) or internally generated (e.g., clock time, successful or unsuccessful execution of another query, results of another query, etc). Trigger queries are provided by declaring the stimulus for a predefined query to be the execution of another predefined query. Real-time queries are established by declaring the stimulus for a predefined query to be internal clock intervals (e.g., once every second). Alerting is accommodated by declaring the stimulus for the predefined query that determines if the alert condition exists to be either clock-time intervals or execution of any queries that can affect alert conditions, whichever is applicable.

Display Formats - The description of display formats is specified by DIAM descriptions, extended to allow descriptions of two-dimensional displacements,[18] stored as relations in the data directory. The information structure over which descriptions are made is the relations resulting from the corresponding RIPL query. Thus, the display of a query may be changed without affecting the query itself.

The description of user-supplied data is also specified by DIAM descriptions. The information structure over which they are declared is the relations that describe the partially predefined query, and the DIAM descriptions need only specify the external representation of attributes that are subject to user specifications.

Integrity Assertions - Integrity assertions are defined in RIPL predicates in terms of the applicable relations and attributes, and are stored in the data dictionary. Assertions range from simple attribute value range

18. C. R. Spath and L. S. Schneider: "A Generalized End-User Facility for Relational Database Systems," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

integrity (e.g., value of employee salary must be between $10K and $30K) to complex aggregate value specification (e.g., the sum of salaries of any department must not exceed 20% of the sum of all departments). Set operations are declared in terms of linking predicates. Thus, the value of DEPT-NO of any employee in EMP must be in the set of DEPT-NOs in the DEPT relation. Literal sets may be declared--EMP/SEX must be in the set MALE, FEMALE.

In general, integrity assertions may include the entire range of RIPL queries. The assertions are processed by the GEUF as described under Generalized End-User Facility (GEUF).

Authorization Constraints - Authorization constraints are defined as RIPL predicates in terms of applicable relations and/or attributes in Boolean combinations, and are stored in the data dictionary. Authorizations may be declared for particular users or sources (i.e., terminals, etc) or both. In general, authorization constraints may include restrictions on any information that can be declared by RIPL statements. Authorization constraints are processed by the GEUF as described under Generalized End-User Facility (GEUF).

Derivations - Derivations are the declaration of named concepts derivable from stored data, stored algorithms, or both. The purpose of derivations is to provide the user with defined concepts that he can reference in RIPL queries without having to derive them independently. Derivations are expressed as RIPL statements over the information structure or other derivations only, and are stored in the data dictionary. The use of derivations extends the user's view of the information structure, allowing him to write simpler queries (RIPL_n) without regard to how the data are actually derived. The GEUF reduces the RIPL_n queries to RIPL_o queries as described under Generalized End-User Facility (GEUF). Any RIPL query that can be expressed in terms of the IS and other derivations may be declared as a derivation and thereafter be viewed as a relation or attribute of a relation.

For example, given the relations

EMP(E#,SAL,DEPT#)

DEPT(DEPT#,NAME,LOC)

representing stored data, and

SUM(A',SUM-A)

representing the algorithm described earlier, users can derive the concept "salary of departments," meaning the sum of salaries of all employees in each department as

(1) GET S(DEPT) .OF. EMP/SAL .WHERE. DEPT#=DEPT/DEPT#

(2) GET DEPT-SAL(DEPT) .OF. SUM/SUM-A .WHERE. A'=S(DEPT)/SAL

The derivation in (1) specifies the set (actually, bag) of salaries for the EMP relation for each corresponding tuple in the DEPT relation, and in (2) specifies the sum of each set of salaries in the same context.

By declaring the above as a derivation in the DD/D, the DEPT relation is extended to include the derived department-salary as

DEPT(DEPT#,NAME,LOC,DEPT-SAL)

Now, users may express queries in $RIPL_n$ to reference the derived attribute as simply

GET S .OF. DEPT/DEPT-SAL .WHERE. ...

The RIPL preprocessor (see page 22) reduces such queries to $RIPL_o$ by substituting the derivation statements (because they are already in $RIPL_o$) for references to derived concepts, appending any user-supplied qualifications as applicable.

Derivations may also be used in qualifications. Thus, to find all departments whose salary is greater than X

GET S .OF. DEPT/DEPT# .WHERE. DEPT-SAL > 'X'

which is similarly reduced to $RIPL_o$.

The example illustrates the power of derivations in simplifying knowledge concepts for users while at the same time making the derivation of the concept visible to management because it is stored in the DD/D.

This concept is extended to provide a subset of natural language by allowing the derivation of relational operators. For example, given the relations

EMP(E#,D#,SAL, ...)

DEPT(D#,NAME,LOC, ...)

SALES(D#,PART,QTY, ...)

find the location of all departments that sell bolts.

The query can be stated in $RIPL_o$ as

GET S .OF. SALES/D# .WHERE. PART='BOLT'

PRINT T .OF. DEPT/LOC .WHERE. D#=S/D#

However, we can derive the concept 'sells' as the second-order relation

SELLS(D#,PART')

by the statement

SELLS(DEPT)=SALES/PART .WHERE. D#=DEPT/D#

declared as a derivation in the DD/D. $RIPL_n$ now allows the same query to be stated more naturally as

PRINT S .OF. DEPT/LOC .WHERE. D# SELLS 'BOLT'

Again, the RIPL preprocessor reduces the query to $RIPL_o$ statements, using the derivation in the reduction.

User Views - User views are the declaration of alternative views of the extended information structure. User views may declare synonyms for concepts (relations, attributes), restrict relations, limit projections of relations, and may declare unnormalized relations for retrievals. The purpose of user views is to allow users to state simpler queries by presenting them with a view of the information structure tailored to their special interests in their vernacular.

User views are declared by RIPL predicates over the information structure, derivations, or other user views and may be declared for a particular set of users or sources. Queries stated over user views $\left(\text{RIPL}_n\right)$ are reduced to queries over the basic information structure only by the GEUF as described under Generalized End-User Facility (GEUF).

Query Compiler/Translator (QC/T)

The QC/T[19] allows users to interact with the entire data and algorithm resources of a computer network as though it were a single integrated system when, in fact, it is distributed as a number of independent and dissimilarly implemented systems. The QC/T accepts RIPL_o queries and generates the required access programs automatically, as well as the logic to synthesize the data returned by each pertinent database into the response sought. The process is shown in Figure 6 and described below.

1) Decompose the query according to the properties of the relational model into subqueries so narrow in scope that no more than one collocated homogeneous* system is necessary to resolve each concurrently, determining the logic to recompose the result into the third-normal-form (TNF) relations defined by the query;

2) Compile each subquery according to the properties of the corresponding DIAM into an access subprogram that is semantically compatible with the pertinent database and algorithms;

---

19. L. S. Schneider: "A Relational Query Compiler for Distributed Heterogeneous Databases," Submitted for publication in *ACM Transactions on Database Systems*, January 1977.

* Because the essence of this process is to transform references to distributed heterogeneous information systems (which we can't process) into references that don't involve distributed heterogeneous information systems (which we can process), we need a term to describe an information system that isn't distributed heterogeneous. The opposite of heterogeneous is obviously homogeneous (similarly implemented). The traditional antonym for distributed is centralized, but this carries the wrong connotation for our use in that there is no central node in a distributed system. The real meaning we want to convey is "stored together" or "co-located," for which there is already an acceptable English word--collocated. Hence, the opposite of distributed heterogeneous for our purpose is collocated homogeneous.

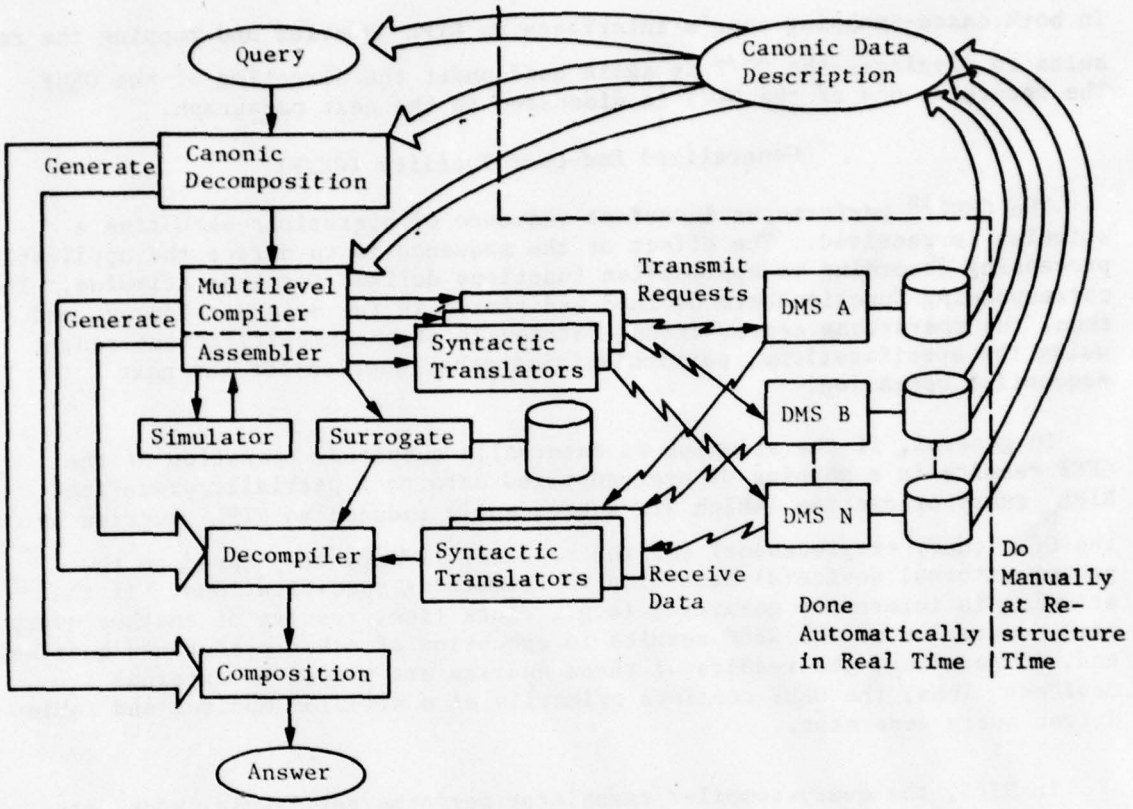


Figure 6. Distributed heterogeneous database query compiler.

3) Recompose the access subprograms according to the DIAMs into the most comprehensive program possible for each pertinent system (concurrently determining the logic to decompose the resulting data into the TNF relations);

4) Perform syntactic translation of each subquery according to the target DMS and transmit according to the protocol of the communications system.

The QC/T contains logic for selecting the most efficient access path (with respect to total network resource use) as described in Reference 19.

The GEUF provides the flexibility in how queries are stated by users, reducing each to $RIPL_o$ before execution by the QC/T. The resulting received data are compiled by the QC/T in a predetermined temporary format, and the details of how it is to be displayed to the user are provided by the GEUF.

19. L. S. Schneider: "A Relational Query Compiler for Distributed Heterogeneous Databases," Submitted for publication in *ACM Transactions on Database Systems*, January 1977.

In both cases--mapping user's interfaces to $RIPL_o$ queries and mapping the results to displays--the QC/T is again used under the direction of the GEUF. The recursive use of the QC/T is discussed in the next paragraph.

Generalized End-User Facility (GEUF)

The GEUF[18] performs an invariant sequence of operations each time a stimulus is received. The effect of the sequence is to direct the applicable processing according to application functions defined for that stimulus. The corresponding functions are defined and stored in the data dictionary, and thus, the operations center around retrievals of the specifications and, using the specifications, particularizing other queries for the next sequential operation.

In general, if the stimulus is externally supplied, operation of the GEUF results in a mapping of user-supplied data to a partially predefined $RIPL_n$ query or queries (which are subsequently reduced to $RIPL_o$ queries by the GEUF query preprocessor) and the results, if any, are mapped to the proper external device(s) according to formatting specifications. If the stimulus is internally generated (e.g., clock time, results of another query, etc), operation of the GEUF results in execution of other predefined queries and, if required, the results of these queries are mapped to external devices. Thus, the GEUF consists primarily of a stimulus monitor and table-driven query generator.

In RIPS, the query compiler/translator performs retrievals, adds, etc to a database by formulating a program in the language of the DBMS, it performs the same functions when the database is the user's terminal in much the same way--by formulating a program in the language of the device driver. Thus, *retrievals* (of user-supplied data), *changes* (to particularize a partially predefined query), and *adds* (to display the results) are directed by the GEUF, compiled into executable programs by the query compiler/translator, but performed by the DMS's and device drivers. The recursive use of the query compiler/translator requires only that the GEUF formulate proper $RIPL_o$ retrievals, changes, adds, and deletes, submitting them to the query compiler/translator for execution. This process is shown in Figure 7.

The GEUF RIPL preprocessor modifies user's queries by appending the integrity and authorization assertions in a similar manner. The GEUF submits retrieval queries to obtain the applicable predicates from the DD/D and change queries to modify the query. The concepts of query modification are taken from those proposed in Reference 20. Similarly, $RIPL_n$ queries are

---

18. C. R. Spath and L. S. Schneider: "A Generalized End-User Facility for Relational Database Systems," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

20. M. Stonebraker: "Implementation of Integrity Constraints and Views by Query Modification," *Proc. ACM SIGMOD International Conference on Management of Data*, San Jose, California, May 1976, pp 65-78 (ed. W. F. King).

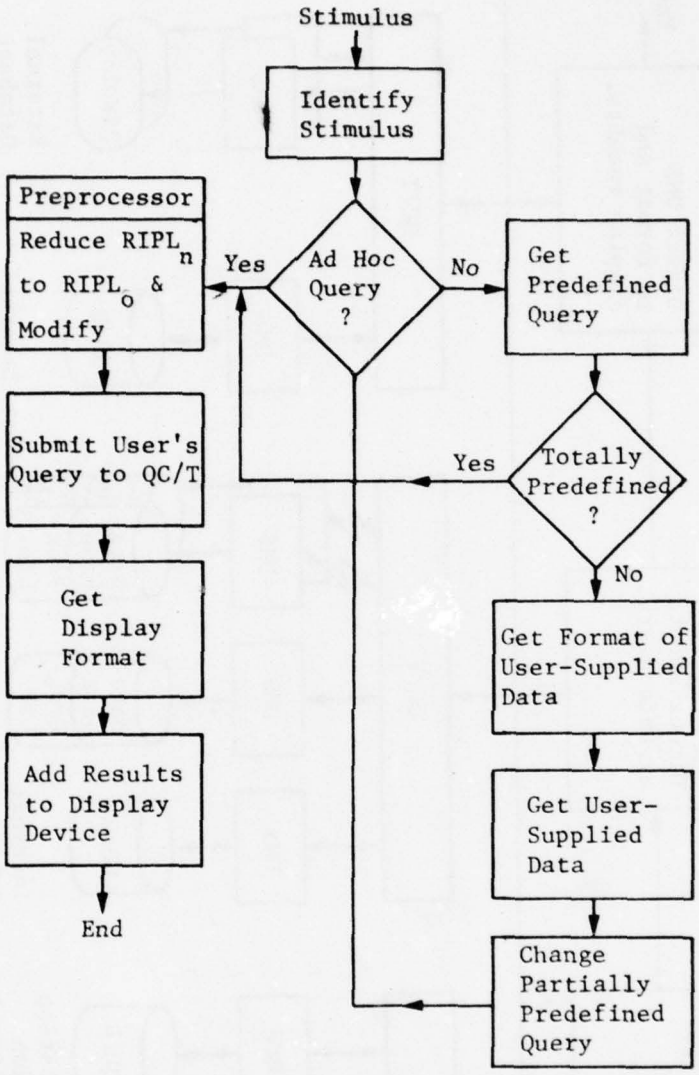


Figure 7. GEUF process.

reduced to RIPL$_o$ by retrieving the applicable declarations and substituting them in place of RIPL$_n$ references, maintaining all original qualifiers.

Functional View of RIPS

In Figure 8, we present a more complete view of RIPS in a distributed information system environment than that shown in Figure 5. Figure 8 shows the concepts of viewing user's devices and the DD/D as part of the database. Just as distributed systems or nodes of the network may be under the control of dissimilar DMSs, so may users' devices be under the control

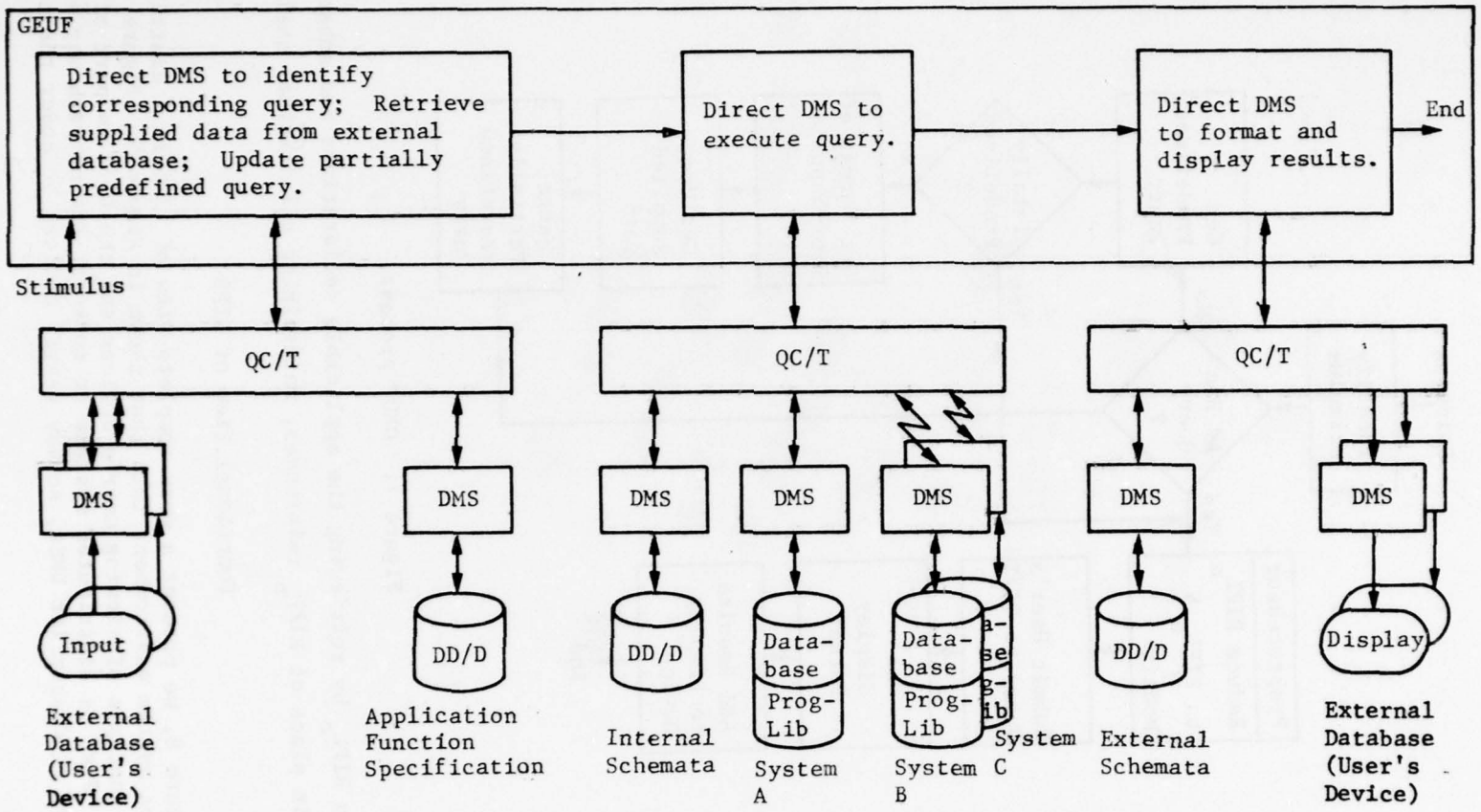


Figure 8. Functional view of RIPS.

of different interfacing software or device drivers. Note that the DD/D is shown to be under the control of a single local DMS, but it too may be distributed.

The sequence of events peformed by the GEUF for each stimulus is shown grouped into three functional blocks. The first block accepts $RIPL_n$ queries or completes partially predefined queries by appending user-supplied data, and reduces them to $RIPL_o$. The second block directs the execution of $RIPL_o$ queries, and the third performs display formatting functions. Reference 18 has a more detailed discussion of the GEUF sequence of operations.

Database Management System Simulator

The DBMS simulator comprises two parts: a math model simulator[6,7] and a real-time simulator[8] as described below.

Math Model Simulator - The DBMS simulator performs a discrete-event simulation of the functions of the general class of database management systems. To represent a broad class of such DMSs, the simulator is based on an underlying canonic model--the Data-Independent Accessing Model I (DIAM I). It was conceived as a tool to aid the study and application of DMSs and allows the simulation of:

1) A user's application in terms of its information structure and traffic rates;

2) A candidate implementation of the application in a DMS, reflecting the proposed implementation of data relationships and recognizing any restrictions imposed by a specific DMS;

3) A candidate host system representing pertinent aspects of the planned host computer and its operating system.

---

18. C. R. Spath and L. S. Schneider: "A Generalized End-User Facility for Relational Database Systems," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

6. L. S. Schneider and T. W. Connolly: "Generalized Data Base Management System Simulator," *Proc 1976 Winter Simulation Conference*, Vol 2, December 1976 (ed. H. J. Highland, et al.).

7. Martin Marietta Database Research Project: *GDMS Math Model Simulator, Functional Specification, Design Specification and User's Guide*. NASA Contract Documentation, NAS9-13951, Johnson Space Center, Houston, Texas, September 1975.

8. Martin Marietta Database Research Project: *GDMS Real-Time Simulator, Functional Specification, Design Specification, and User's Guide*. NASA Contract Documentation, NAS9-13951, Johnson Space Center, Houston, Texas, September 1975.

Use of the simulator permits studies relevant to the device and use of DMSs. Typical studies of this nature might include:

1) Comparison of the operating performance of two competing DMSs for the same application;

2) Comparison of the operating performance of various implementations of an application using options available in a single DMS;

3) Comparison of operating results based on differing host-system configurations being considered;

4) Studies to enhance the user's knowledge and familiarity with DMS techniques.

The simulator consists of subsystems, each hierarchically subdivided into a number of modules. Four of the principal correspond generally to the four subsystem levels of the DIAM:

1) Information-based model - Generates queries representing the application under study and maintains data population statistics;

2) Structure-based model - Accepts the queries as Representation-Independent Accessing Language (RIAL) statements and uses definitions of implemented access paths to produce Representation-Dependent Accessing Language (RDAL) statements;

3) Procedure-based model - Accepts the RDAL and produces a sequence of input/output accesses based on the definition of how and where access paths and data are stored;

4) Host model - Represents host computer-system logic, including its peripherals and operating system, as it pertairs to calculation of response time and resource use in processing I/O access resquests.

An executive subsystem accepts control from the operating system at execution time, and contains modules to read and store simulation data, configure the simulation, control experimental runs, and produce the required output.

The entire simulator has been programmed in the FORTRAN language with very few deviations from ANSI FORTRAN standards. The source code has been implemented on CDC 6000, Univac 1100, and IBM 370 systems operating in a batch environment.

Real-Time Simulator (RTS) - The real-time simulator is designed to provide empirical baseline data to support simulation experiments conducted with the MMS. The primary need for these data is to support calculation of the MMS in new experimental situations, particularly where empirically derived analytic functions are being employed in the predictive process. This can be effectively satisfied by a GDMS "test bed" in which stimulus can be controlled and resulting performance measured.

Principal software components of the real-time simulator include:

1) The Database Manager - A generalized data management system that is a candidate being evaluated for a given application;

2) An Input Load Unit - A program component that controls (or generates) a traffic load for processing by the DBMS. When the actual database for the application under study does not exist, significantly distributed symbolic instances are automatically generated in accordance with the QDD static descriptions. Similarly, transactions described by the QDD dynamics can be generated, ensuring that the application descriptions for both the math-model and real-time simulations correspond.

3) An Instrumentation Unit - A program component that measures and reports the time and resources used by the host system for processing each item of traffic (assumed to be vendor supplied);

4) An Anlysis Unit - A program component that controls the experimental process in the RTS and produces the required outputs for external use.

The real-time simulator resides in the actual host computer and operating system, and uses an actual GDMS together with input, instrumentation, and analysis units. The resulting implemented test bed permits measurement and evaluation of the GDMS under actual working conditions. Instrumentation results provide values of environmental and functional parameters that correspond to those used in the math-model simulator.

DMS Software Evaluation Methodology

The purpose of the DBMS math-model and real-time simulators described in the previous section is evaluation of software for quantifiable performance characteristics. However, evaluation of software extends to characteristics that are inherently unquantifiable. These include ease of use, conformance to standards, vendor support, documentation, data independence, and others.

The use of simulation imposes a formal approach to requirements definition and performance analysis for quantifiable characteristics, and the DMS software evaluation methodology extends this approach to include the equally important unquantifiable issues. The essence of this methodology is to establish such issues as constraints in the selection process and to either eliminate candidates that are unable to satisfy these constraints or to derive the cost of satisfying the constraints and adding these to the life-cycle cost profile.

Thus, for example, if vendor support is required, we must establish a level of such support and obtain a commitment from candidate vendors. If a required standard is not met by a candidate package or design, we must obtain a cost for bringing it into conformance or eliminate it from further consideration.

The objective of this approach is to compare candidates objectively on an equal basis for their ability to satisfy the requirements and to eliminate, as much as possible, the subjectivity involved in using such techniques as rankings by weighted scores. This provides candidate vendors or designers the opportunity to bring their products into compliance, where practical, and reduces the selection criterion to one of life-cycle cost.

SYNOPSIS OF KM REQUIREMENTS

This section lists KM requirements in Section 2 of Refernce 1 and in the appendix of Reference 2, taken in their order of appearance. References are made to their appearance in the source documents, and cross-referenced to the discussion of their allocation to RIPS components in this document. Additional KM requirements are discussed in the following section in terms of the KM logical system design.

No.	Requirement	Source & page ref	Discussion page ref
1.	EA will require . . . a powerful KM facility to keep track of the many data elements, file structures, databases, and flows that compose the knowledge of the corporation.	1. p 33 2. p 108	38
2.	The EA should prepare some sort of model of the relationship of the enterprise to other organizations, including the volume, direction, and importance of various data and information flows. Similarly, the EA should develop a model of the macroview of the enterprise itself . . . including tasking requirements, data, and information flows.	1. p 33 2. p 108	38
3.	Selection guidelines will need to be established for all data management software.	1. p 34 2. p 109	43
4.	Evaluation procedures, sample databases, benchmark tests, and checks on the consistency of a proposed system . . . will need to be developed.	1. p 34 2. p 109	43

---

1. James F. Berry and Craig M. Cook: *Managing Knowledge as a Corporate Resource*. Contract Source Document, Version 4.5, 28 May 1976.

2. James F. Berry and Craig M. Cook: *Viewing Knowledge as a Resource in Federal Departments of the U.S. Government*. Economic Research Service, U.S. Department of Agriculture, September 1977.

No.	Requirement	Source & page ref	Discussion page ref
5.	Production of useful guidelines for preparation of meaningful impact studies is an area needing further research.	1. p 34 2. p 109	43
6.	. . . costs of standardization must not be transferred to database end users by requiring them all to view data in exactly the same way. . . . On the contrary, the goal of standardization should be to facilitate communication among different divisions of the enterprise by agreeing on standardized concepts--not standardized names.	1. p 35 2. p 109	38
7.	. . . the Knowledge Resource Center (KRC) . . . should contain summary information about the other databases of the enterprise . . . driven automatically by information from the other databases . . . . It should provide special user interfaces . . . for handling the kinds of questions that top management asks.	1. p 36 2. p 111	38
8.	An integral part of the KRC . . . is a data dictionary/directory.	1. p 37 2. p 111	38
9.	In the area of security, the EA must . . . take extraordinary measures to preserve the integrity and privacy of . . . metadata.	1. p 38 2. p 112	38
10.	There is great need for tools for hardware/software tuning, schema design . . ., models of significant performance variables, . . . to measure actual database use and compare it against original design specifications to determine when restructuring is warranted.	1. p 39 2. p 113	43
11.	A dynamic restructuring capability . . . can be a significant performance factor.	1. p 39 2. p 113	49
12.	There is a need to develop an evaluation procedure for determining the suitability of various access methods for different applications.	1. p 39 2. p 113	43

1. James F. Berry and Craig M. Cook: *Managing Knowledge as a Corporate Resource.* Contract Source Document, Version 4.5, 28 May 1976.

2. James F. Berry and Craig M. Cook: *Viewing Knowledge as a Resource in Federal Departments of the U.S. Government.* Economic Research Service, U.S. Department of Agriculture, September 1977.

No.	Requirement	Source & page ref	Discussion page ref
13.	In the security area, there is a great need for improved security techniques in all layers of hardware/software.	1. p 39 2. p 114	51
14.	Devise a workable and agreed-upon technique for locking and for resolving deadlocks.	1. p 40	50
15.	Rapid recovery from failure (is required).	1. p 40 2. p 114	33
16.	Adequate audit-trail capabilities for back-up, integrity checking routines for maintenance, and restoration tools for recovery need to be developed.	1. p 40 2. p 114	33
17.	A technique for automatically checking the semantic consistency of data.	1. p 40 2. p 114	38
18.	Tools are needed to improve the . . . process . . . of the physical mapping of the logical database to physical devices.	1. p 40 2. p 114	43
19.	A methodology is needed for determining in advance the expected size of a database as well as performance characteristics.	1. p 40 2. p 114	43
20.	Models are necessary to allow simulation of the effects of certain parameter changes on the performance of the system.	1. p 40 2. p 114	43
21.	Schema navigation tools are needed to assist the DBA in purusing and altering existing schemata.	1. p 40 2. p 114	38
22.	Automated procedures for migrating existing databases to new hardware or software are essential.	1. p 40 2. p 114	38
23.	Techniques are needed for automatically generating schemata from diagrams or sample programs.	1. p 40 2. p 114	49
24.	A method for keeping track of multiple versions of a schema would assist in maintaining a database whose structure changes dynamically.	1. p 40 2. p 114	38

1. James F. Berry and Craig M. Cook: *Managing Knowledge as a Corporate Resource*. Contract Source Document, Version 4.5, 28 May 1976.

2. James F. Berry and Craig M. Cook: *Viewing Knowledge as a Resource in Federal Departments of the U.S. Government*. Economic Research Service, U.S. Department of Agriculture, September 1977.

No.	Requirement	Source & page ref	Discussion page ref
25.	Develop a procedure for determining if the internal schema as designed meets the user's or AE's requirements.	1. p 40 2. p 114	43
26.	Techniques for easing data migration or roll-over of application from one system to another.	1. p 41 2. p 115	38
27.	Devise an effective scheme for keeping copies of a database is synchrony.	1. p 41 2. p 115	38
28.	Develop ways of dynamically allocating network resources (e.g., storage, communication facilities, data management capabilities, etc).	1. p 41 2. p 115	48
29.	Develop and employ subnetwork models within a computer network.	1. p 41 2. p 115	38
30.	Provide local and global views of a database to enhance performance.	1. p 41 2. p 115	38
31.	Research needs to be done on how to allow the user to make an easy transition from one interface to another.	1. p 42 2. p 115	38
32.	Increase the effectiveness of data presentation . . . such as superimposing images over pictures (e.g., slides or television).	1. p 42 2. p 116	38
33.	An automatic exception-reporting capability in which an alerter is triggered when certain user-specified conditions occur.	1. p 42	38
34.	Methods of summarizing data from numbers, graphs, or text, and presenting summaries need to be investigated.	1. p 42 2. p 116	51
35.	End users and AEs have a need for a navigation facility which will allow them to browse through a database with an unknown schema.	1. p 42 2. p 116	38
36.	The database must be capable of instructing the user as to its structure and use.	1. p 42 2. p 116	38
37.	Techniques are needed for determining the 'optimal' path to a data item.	1. p 42 2. p 116	38

1. James F. Berry and Craig M. Cook: *Managing Knowledge as a Corporate Resource*. Contract Source Document, Version 4.5, 28 May 1976.

2. James F. Berry and Craig M. Cook: *Viewing Knowledge as a Resource in Federal Departments of the U.S. Government*. Economic Research Service, U.S. Department of Agriculture, September 1977.

No.	Requirement	Source & page ref	Discussion page ref
38.	Methods are needed for automatically generating the necessary access code.	1. p 42 2. p 116	 38
39.	Preretrieval query analysis and simulation can help conserve machine as well as human resources.	1. p 42 2. p 116	38 43
40.	The techniques of computer-aided instruction should be applied to the task of informing the user how to best use the resources available.	1. p 42 2. p 116	 38
41.	Initial validation of input data must occur before the data are entered into the database . . . using methods of error detection and correction . . .. Once the data are entered, further validation should be performed as an integrity check.	1. p 42 2. p 116	
42.	A scheme is needed to validate derived data or the algorithm used.	1. p 43 2. p 116	 59
43.	Validation techniques are needed for testing the consistency of the conceptual, internal, and external schemata.	1. p 43 2. p 116	 43
44.	A method of associating a validity value with each data element and with databases in general needs to be developed so that meaningful validities can be assigned to information derived from multiple sources.	1. p 43 2. p 116	 45
45.	The capability to check context integrity before releasing information is needed.	1. p 43 2. p 117	 38
46.	The ability to validate queries before they are executed.	1. p 43 2. p 117	 47
47.	Efficient techniques for handling the entry of large volumes of data are needed.	1. p 43 2. p 117	 43
48.	Standard data entry techniques would reduce training and increase efficiency.	1. p 43 2. p 117	 38
49.	Tools are needed to model the real world, to develop and test hypotheses about the real world based on the data available, and to project the implications of a hypothesis about the real world through some simulation mechanism.	1. p 43 2. p 117	 38

1. James F. Berry and Craig M. Cook: *Managing Knowledge as a Corporate Resource*. Contract Source Document, Version 4.5, 28 May 1976.

2. James F. Berry and Craig M. Cook: *Viewing Knowledge as a Resource in Federal Departments of the U.S. Government*. Economic Research Service, U.S. Department of Agriculture, September 1977.

No.	Requirement	Source & page ref	Discussion page ref
50.	Improvement is needed in question presentation, interactive question enhancement, application of probablistic rules (fuzzy logic), basic inferencing techniques, answer presentation, proof demonstration, and inductive inferencing of rules from sample data (with application to trend analysis).	1. p 44 2. p 117	45
51.	The use of time dependencies on queries in order to keep straight the accession of archival databases.	1. p 44 2. p 117	44

KM LOGICAL SYSTEM DESIGN CORRELATION TO RIPS

In this section, we discuss the RIPS concepts as related to the KM logical system design described in Reference 2. The requirements of each subsystem of the KM design are addressed, along with the requirements listed in the preceding section. The subsystems proposed by KM are the Factual Knowledge Subsystem, Procedural Knowledge Subsystem, Judgment Support Subsystem, and the Translation and Control Subsystem. In addition, we include discussion of a simulation subsystem and its relationship to the others to address the KM requirements for performance analysis.

Factual Knowledge Subsystem (FKS)

The FKS (Data Management Subsystem in Reference 1) is viewed in the KM concept as the DBMS software or access engines. In RIPS, the query-compiler algorithm can traverse a database schema specified by DIAM description and, thus, if the results of a query compilation were machine-language I/O instructions, the query compiler would serve as an access engine. However, in the environment intended, the results of a query compilation are translated into the language of whatever DBMS or access engine is implemented. In RIPS, functions required of the KM Data Management Subsystem are performed by existing DMSs that comprise the nodes of the information system network, and the RIPS query compiler/translator is functionally a part of the KM Translation and Control Subsystem's logical design. RIPS operates in a distributed heterogeneous database environment and imposes no requirements on existing systems to be brought into compliance with some arbitrary standardized implementation. Conceptually,

---

1. James F. Berry and Craig M. Cook: *Managing Knowledge as a Corporate Resource*. Contract Source Document, Version 4.5, 28 May 1976.

2. James F. Berry and Craig M. Cook: *Viewing Knowledge as a Resource in Federal Departments of the U.S. Government*. Economic Research Service, U.S. Department of Agriculture, September 1977.

a RIPS package can be installed at one or more nodes in the network without changing the existing implementation, allowing access to distributed data and processes that require it without the user's regard as to how or where they are implemented.

Rapid recovery from failure (R-15) is primarily a requirement of the DMS installed at each node in the network, as are database restoration (R-16) and reorganization. However, the RIPS must maintain a transaction log in case some system in the node discovers that erroneous data may have been supplied to previous queries. In such cases, the RIPS must perform a corresponding recovery and restoration with respect to the queries it has originated.

An operating system-supplied file structure that will allow users to create and keep their own personal files, which may not be part of the knowledge resource, is required by KM. While RIPS accommodates this capability technically, some of the power of the KM concept will be jeopardized. In RIPS, the database is characterized by a profile of the data and processing requirements via the quantitative data descriptions in the DD/D. Because the QDD is the one source of information regarding performance analyses, organizational information flows, etc; any implementations not recorded will not be included in the analyses, reducing their fidelity. If the QDD profiles are included in the DD/D, the corresponding data automatically become part of the knowledge resource, and even though the data may not require active management of the EA, the data will require active management by the DBA. However, the extent to which QDD profiles are maintained is an installation-dependent decision, and whatever fidelity is justified can be accommodated.

A text-editor user interface is required in KM and is discussed under Text Processing.

Procedural Knowledge Subsystem (PKS)

The PKS's task is to manage the procedural knowledge of the knowledge resource. In RIPS, this function is provided by representing algorithms and application programs of an existing system as relations in the IS, accessible through RIPL, tailored to end users by the GEUF. Details of where the algorithms are located and how they are executed are described in the data directory and are used by the QC/T in formulating programs for execution and, at the same time, are visible for management. Functionally, the resulting program doesn't differ from a program generated as in the previous paragraph, and thus, the program is executed by existing DMSs that comprise the nodes of the information system network, and the RIPS QC/T is functionally part of the KM Translation and Control Subsystem's logical design.

The heuristic component of the KM PKS is described as existing knowledge-based systems. Interfacing with such systems in RIPS should present no additional problems. At the information level, RIPS does not distinguish between the originally intended purpose of a resource in the network, nor its method of implementation. Rather it concentrates on

information available from the node that is to be made accessible to other nodes. The knowledge-based systems listed in Reference 2 have one thing in common--a database and products generated from the application of rules and algorithms (which are also data in some context) over the database. Either the final products themselves or some portion of the contents of the database used in these productions, or both, need to be accessed by other nodes. Otherwise, there is no reason to include such systems in the network. Whatever information is to be made available is represented in the RIPS information structure. For example, if only the products of an existing system are to be included, a single relation may suffice, as perhaps

PRODUCT (PROD-ID, TIME, ...)

and suitable partially predefined query(s) and external interfaces can be tailored to the using environment. Of course, details of where the application is implemented, how it is executed, and the translation descriptions are entered in the DD/D.

If only part of the data used in the production is to be made accessible, then only a description fo the data need be included in the Information Structure, for example

TABLE-NAME (ID, V_1,V_2, ...)

and the corresponding supporting descriptions entered in the DD/D.

However, the real justification of the KM concept is the more difficult case in which data or applications from one node are needed in conjunction with data or applications from others to satisfy a single query. It is this environment that RIPS envisions.

A basic precept in the RIPS philosophy is that the semantics of functions must be separated from the implementation details to make the knowledge widely available and manageable. This includes stored data, algorithms, application programs, or entire systems. The degree to which an organization's knowledge is to be made available is properly the subject of organizational management, and whatever choice is made must be supported technically by RIPS.

Judgement Support Subsystem (JSS)

The KM JSS (User Interface Subsystem in Reference 1) logical system design requires flexible user-query-statement and data display techniques. The flexibility is provided in RIPS by the Generalized End-User Facility, which directs the mapping of external representations to queries in terms

---

2. James F. Berry and Craig M. Cook: *Viewing Knowledge as a Resource in Federal Departments of the U.S. Government*. Economic Research Service, U.S. Department of Agriculture, September 1977.

1. James F. Berry and Craig M. Cook: *Managing Knowledge as a Corporate Resource*. Contract Source Document, Version 4.5, 28 May 1976.

of the information structure and mapping of the results of queries to external representations or displays. Thus major functions required in the KM logical design are allocated to the KM Translation and Control Subsystem discussed in the next section.

In RIPS, there is a high degree of symmetry between the functions of database and end-user device management. By viewing the end-user device as a database, data stored in it at any particular moment can be described in the same terms (DIAM descriptions) as data stored in the internal database. Because the process of reading from, or writing to, user's devices is essentially the same as reading from or writing to database storage devices, the algorithm that traverses the DIAM descriptions of data storage implementations can also traverse the DIAM descriptions of display formats. In the RIPS view, a user's retrieval query from the internal database is simultaneously an add query to the external database or user's device. This concept, discussed in Reference 18, is summarized below.

When the query compiler receives a retrieval query, it must determine how and where the required data are stored. This information is provided by DIAM descriptions stored in the data directory. The query compiler uses these descriptions to formulate a program to retrieve the data. The program is translated into the language required by the DMS that controls the database(s), and the data are returned to the compiler, which then assembles the data into a temporary storage. Because the result of each RIPL statement is a relation, the temporary storage contains the relations derived by the query. Thus, a RIPL query contains, implicitly, the relational or conceptual view of the results.

Now the reverse process must take place. Relations generated by the query must be added to the external database. The GEUF automatically generates the ADD query and submits it to the query compiler. When the compiler receives an add query, it must determine how and where the required data are stored. This information is the formatting specifications supplied by the user as DIAM descriptions and stored in the data directory. The query compiler uses these descriptions to formulate a program to store (display) the data accordingly. The program is translated into the language required by the device driver or operating system that controls the device.

In RIPS, this concept is extended to map user-supplied data via whatever interfacing technique is desired (forms, menus, light pen, etc) to partially predefined RIPL queries.

The symmetric view of the internal and external mappings allows recursive use of the query compiler/translator software and thus provides end-user interface independence at the external level just as it provides database implementation independence at the internal level.

---

18. C. R. Spath and L. S. Schneider: "A Generalized End-User Facility for Relational Database Systems," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

The requirement to provide a set of common utilities, such as sorting, report generation, etc, accessible by the user is implicit in the RIPS concept, and makes the use of such utilities clear to users by requiring only that the format of displays be described--not the procedures for how the formats are to be generated.

The KM requirement for a "universal interface," providing the user with a common language for selecting various interfaces, is described in the next section, as are the eleven interfacing techniques specified by the KM logical system design.

In addition, a Knowledge-Based Personal Assistant (KBPA) is required to aid users by providing substantial knowledge of what a particular user needs to do his or her job. In RIPS, this requirement is satisfied by the GEUF and the concepts of derivations, user views, and partially predefined queries, as discussed below.

In an organizational environment, individual users of the information system are not free to perform operations over the information at will. Their use of the system is constrained by the purpose of their job, just as their use of physical resources is. However, the use of automated information has added, or at least changed, tasks that are necessary for their job; that is, they must query or update databases. The degree to which this constrained interaction corresponds semantically with their job largely determines the success of the system.

For example, a motel reservation clerk in New York, when asked to reserve a room at the Downtown Atlanta Motel, performs the task by making an entry via a terminal. The transaction is stated in terms of reserving a room. It is not viewed as updating the database, even though that is precisely what is being done and may, in addition, require computational or other algorithms (i.e., internal knowledge) to determine whether a room is available, of which the clerk is totally unaware.

Well-designed user languages make extensive use of verbs in the vernacular of the user community for the dual purpose of aligning the semantics with the job and constraining the operations to just those required. They do this by providing application programs that recognize only these functions, and translating the requests into database and algorithmic operations. Thus, while the motel clerk may reserve a room in Atlanta, he cannot assign a particular room, add new rooms to the motel, or determine who is in a particular room. But these operations are meaningful and required for some users in the network.

In RIPS, both the semantics and the constraints are implemented through the use of partially predefined queries, derivations, and user views. Whatever knowledge or rules are to be applied are specified as derivations (for general use) or in the partially predefined queries (for particular use). Because they are stored in the DD/D, their use is controlled through specification of authorization constraints. The interface, or user's language, to these partially predefined queries is tailored by the

declaration of formats to require only user-supplied values to particularize the query for the current task. This includes whatever external representations are called for in the vernacular and in the users' environments.

Today, application programs can provide whatever interface and constraints are needed, incorporating the knowledge to assist the user. But they are bound to the current database implementation; new information requirements are difficult to accommodate even though the necessary data are already available; and the internal knowledge is not readily accessible to management. In RIPS, the partially predefined queries are not bound to the implementation, either external or internal; new requirements are easily defined as updates to the DD/D via RIPL (or tailored forms); and the internal knowledge is readily accessible through RIPL queries to the DD/D.

Again, this flexibility is in keeping with the RIPS philosophy stated under Procedural Knowledge Subsystem (PKS).

Translation and Control Subsystem (TCS)

Requirements in the logical system design of the Translation and Control Subsystem are allocated to RIPS components as follows. The canonical form for data structures and data formats is satisfied by the RIPS information structure and DIAM descriptions of implementations stored in the DD/D. The mapping mechanism is provided by the query compiler/translator and the RIPL preprocessor, using the specifications stored in the DD/D. The enterprise knowledge resource is satisfied by recursive use of RIPS concepts in the management of the DD/D and distributed data. Tailored user interface techniques are provided by the GEUF, including the RIPL language, RIPL preprocessor, and recursive use of the QC/T. In the following paragraphs, some details are provided for each requirement, including those mentioned in the Introduction.

The conceptual model of the enterprise's data is provided by the RIPS information structure. The canonic model employed in RIPS provides a view of stored data, including data stored in the DD/D, and stored algorithms. In conjunction with the RIPL language, the information structure satisfies R-49 by providing a sufficient model of the real world over which hypotheses can be developed. In this regard, an important RIPS concept is inclusion of QDD parameters in the information model. This provides visibility of the populations; population distribution; and arrival, change, and departure rates of entities in the real world of interest. This model of the dynamics of the real world is valuable knowledge, not only from the standpoint of specific production information processing (e.g., at what rate do competitors enter and leave the fields?) but also in the implementation decisions owing to the dynamics. The adequacy and fidelity of the model and hypotheses can be tested by periodically performing real-world experiments and comparing them with the current information model and QDD statistics.

The information structure is the foundation for satisfying R-26 because both stored data and algorithms are represented, and RIPL queries remain stable under the migration of either.

The requirements for data translation (R-22, R-26) are satisfied through recursive use of the query compiler and GEUF, which together form a data translator. In normal use, a user's retrieval query is simultaneously an add query in which the results are added to an external database (display) according to DIAM descriptions of the format. This symmetry allows the external database to be another internal database because DIAM descriptions and compiler's operations are independent of the device. Only the translator is aware of any difference, and thus, translation of one internal database to another requires only DIAM descriptions of both, and applicable RIPL queries to retrieve whatever portion of the source database is desired with the correlation of whatever target database is required.

The RIPS information structure includes the three-part association of all entity name sets, or in relational terms, the relation/name/domain-name/attribute (role) name association. RIPL queries including adds and changes over the information structure can be automatically checked for semantic consistency (R-17) to the extent that the domain-name/role-name compatibility can be assured. For example, a query that compares the role's age with street number may be numerically legal (i.e., both are integer numbers), but is semantically questionable because they are from different domains. Further checks on context integrity (R-45) are provided by the QC/T query decomposition process. Queries that cannot be decomposed in terms of the information structure are ambiguous.

The RIPS DD/D (R-8) contains all the specifications required by the GEUF and the query compiler/translator, including the information structure, which in turn includes the information structure of the specifications themselves as part of the database. R-1 requirements are satisfied by providing access and management of the "data elements, file structures, databases, and flows that comprise the knowledge of the corporation." General requirements for metadata management are provided by the GEUF, tailoring the interface to whatever technique is called for, and providing access to either metadata or stored data in a consistent manner (R-6).

The integrity and privacy of metadata (R-9) are provided in the same manner and to the same degree as for stored data. Access to the data dictionary provides the tools for the DBA to peruse and alter existing schemata (R-21), and again, tailored user-oriented interfaces are provided in keeping with the DBA's skills and needs. The data directory--describing how and where the data are stored--provides a view of subnetworks within the computer network as required by R-29. Local views of the database (R-30) are implemented by declaring user views and partially predefined queries.

The QDD provides both a macroview of the enterprise (R-2) and a microview (R-29) characterizing the data flow for individual users. A part of the DD/D, the QDD can be maintained to whatever concurrency provides the fidelity required through the use of 'trigger' queries that update the QDD based on receipt and execution of other queries.

Mappings to internal implementations are performed in RIPS by the RIPL query compiler/translator, operating over DIAM descriptions of implementations. DIAM descriptions, stored in the DD/D, also provide the DBA with knowledge and visibility of the various implementations (R-21, R-24) and, in conjunction with the QDD, provide the compiler with accessing and quantification parameters needed for optimization (R-37, R-39). The compiler/translator automatically generates the necessary access code (R-38) for the applicable DMS.

Where data are redundantly stored in the network, the query compiler's optimizer determines which access paths to use for efficient resource use. For additions, deletions, and changes, all instances are maintained, thus satisfying R-27.

The RIPL allows users to state what information is required--not how the data are obtained. Queries remain stable regardless of how or where in the network the required data are implemented, and changes to implementations have no effect on the queries. Queries may include requests for both derived data and stored data in a consistent manner so that a change that replaces stored data with an algorithm for deriving the same data, or vice versa, has no effect on queries.

This implementation independence is provided by the RIPL language and the RIPL query compiler/translator that automatically generates the required accessing programs using the implementation specifications (DIAM descriptions). Implementation descriptions are not restricted to automated data, but extend to descriptions of manually stored data sources. This concept provides a consistent methodology for the KM concept because management of corporate information requires knowledge of the formal lines of communication in the organization, whether automated or not. A query that requires both computer-stored data and manually stored data can only be answered by accessing both, and the methods of accessing the required data must necessarily differ, but the methods of stating the information requirement need not.

Techniques for accessing manually stored data are dictated by the use of the data. If no computations, correlations to computer-stored data, or special report formats are needed, the response to a query could contain only a description of where the manually stored parts of the query can be found (e.g., office number, file name, person, etc). Otherwise, manually stored data comprising the answer must be entered in the computer to produce the final product. This can be accomplished to whatever degree of automation is demanded, including automatically issuing a request to the manual data manager (e.g., clerk, librarian) and accepting the answer via a terminal input, then completing the processing. If manually stored data are subsequently automated, there is no effect on users--queries remain stable and only response time may differ.

GEUF concepts accommodate all types of queries, as described in Reference 18 and summarized below. Ad hoc queries are simply stated in RIPL at the time desired. Totally definable queries that are to be executed at the occurrence of some predetermined event--such as real-time queries--are predefined and stored in the DD/D, along with a description of the initiating stimulus. The GEUF monitors all events that serve as stimuli, and upon receipt, retrieves the corresponding predefined query and submits it to the query compiler to be executed. As part of the database, the DD/D can be implemented by whatever means are required. Retrieval of a predefined query from the DD/D may involve only a main memory access if that is where the query is stored.

Queries that are partially predetermined--either the stimulus or some portion of the query is to be supplied by the user at execution time--are predefined and stored in the DD/D. If the stimulus (e.g., function key, etc) is to be supplied, its description is stored in the DD/D along with the predefined query and the correlation between them. When the GEUF receives the stimulus, it retrieves the corresponding query and submits it to the query compiler for execution.

If a portion of the query is to be supplied by the user at execution time to particularize the context for current needs, the stimulus is the receipt of the user-supplied data by whatever means are appropriate (e.g., forms, including menus, terse command language, etc). The predefined part (partially predefined query) of the query is stored as relations in the DD/D, and the description of the external representations is stored as DIAM descriptions in the DD/D. The GEUF views user-supplied representations as update or change queries to the relations containing the partially predefined query and initiates the change by issuing updating queries to the query compiler/translator, which in turn performs the update. The completed query is then submitted to the compiler/translator for execution.

The technique by which the user interfaces with partially predefined queries includes all techniques required in the KM logical system design of the user interface subsystem. Specifically, the eleven required interfaces are accommodated as follows. A subset of the natural language is accommodated through the use of RIPL ad hoc statements using the concepts of derivations and user views. A graphic representation is definable in DIAM descriptions for either input (via light pen, etc) or display. A forms-oriented interface is provided by DIAM descriptions of whatever geometry is desired interacting with partially predefined queries. Menus are special cases of forms, and entry of a 'selection mode' is simply the representation of either data to complete a partially predefined query or the stimulus to execute a predefined query. Dialogue is provided by repeated use of partially predefined queries in which receipt of a stimulus executes a query that retrieves a form or prompting message, and the

---

18. C. R. Spath and L. S. Schneider: "A Generalized End-User Facility for Relational Database Systems," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

subsequent receipt of the form is the stimulus to execute another predefined query, complete and execute a partially predefined query, or both. A transaction-oriented interface is a special case of forms in which the geometry of the form is a command-like sentence.

A relational interface is provided either by RIPL, or a forms-oriented approach like Query by Example[13] is provided by the forms interface to partially predefined queries so that the partially predefined queries include only the relation name (or range) of a statement, and all other parts of the query (e.g., attribute list, function code, qualifying predicates, etc) are user-supplied at execution time. Access of computational processes, required by the KM programmatic interface, is provided by inclusion of relational descriptions of algorithms in the RIPS information structure and the ability of the compiler/translator to execute the algorithms. The navigational interface is provided by access to either the database or the DD/D by the RIPS concept of viewing the DD/D as part of the database and descriptions of DD/D contents as part of the information structure, thereby allowing use of all RIPS capabilities in the KM environment (R-35). Text editing is discussed under Text Processing.

Transition of one interface to another (R-31) is accomplished by altering DIAM descriptions of user-supplied representations, changing the stimulus definition of predefined queries, or changing device driver specifications for the query compiler/translator. All these, including combinations, preserve the semantics of the underlying query. The ease with which this can be accomplished is apparent from a technical standpoint--the DD/D must be updated accordingly. However, from the user's standpoint, the ease of changing from one interface to another is properly the subject of human factors, including training, environment, expertise, aptitude, etc. However, the RIPS concept separates the semantics from the implementation and includes a wide range of alternatives with relatively small programming effort, allowing these decisions to be made quickly and effectively. Thus, RIPS agrees with the KM assertion that a single user interface is incapable of satisfying user-community needs and provides ease of migration from one to another, technically, leaving the choice of techniques to the human factors discipline (R-32, R-34).

The concept of partially predefined queries satisfies the alerting or trigger query functions required by R-33, and allows flexible composition of whatever dialogue or CAI is required by R-36 and R-40. Standard data entry techniques can be tailored for applicable users to reduce training and increase efficiency, satisfying R-48 and the external considerations of R-47.

---

13. M. Stonebraker, E. Wong, and P. Kreps: "The Design and Implementation of INGRES," *ACM Transactions on Database Systems*, Vol I, No. 3, September 1976, pp 189-222.

Simulation Subsystem

Performance analysis, evaluation, and DMS software selection are integral parts of RIPS. While the DMS simulator is not allocated to any of the three KM subsystems, it does use many RIPS components that are. Specifically, the information structure, QDD, DIAM descriptions, and major functions of the query compiler are all part of the simulator.

In the math-model simulation mode, the information-based model generates simulated time-tagged queries that statistically represent the workload described in the QDD of the application under study. In the KM environment, if the purpose of a performance analysis is to evaluate implementations of the existing workload, the current information structure and QDD consitute the workload and thus become the input to the information-based model of the simulator, and the candidate implementations are described in the lower-level models of the simulator. If the purpose is to evaluate the effect of changing workloads on the existing implementation, the modified QDD (to reflect the new workload) becomes the input to the information-based model, and current implementation descriptions (DIAM descriptions) form the input to the lower levels. If the purpose is to determine the effect of a change to the information structure, then both the modified QDD and candidate implementation descriptions are input to the simulator. If the purpose is to evaluate a different host computer, for the existing implementation, the host computer model of the simulator must be changed accordingly.

In the real-time mode, QDD static descriptions are used to generate statistically significant symbolic instances that the compiler/translator translates into actual update instructions to the DBMS under study. This produces a baseline sample database to provide real-time performance measures. The time-tagged transactions or queries that represent the workload are generated by a query generation module according to the QDD parameters. These are similarly translated into the DML with compatible symbolic values in the qualifiers and submitted to the actual DBMS under control of the simulator. Performance measures are accumulated by a commercial performance measurement program installed in the host computer.

These capabilities satisfy R-4 by providing evaluation techniques, sample databases, benchmarks, and checks on the consistency of a proposed implementation. R-10 is satisfied by allowing modeling of significant performance variables for implementation tuning and for comparing design specifications with actual use. Simulation of the performance of candidate access methods for the specific application described by QDD satisfies R-12, R-18, and R-20, and produces an accurate measure of database size owing to the data and access path overhead before implementation, satisfying R-19 and R-25. R-39 is satisfied in the simulation mode before implementation, and by the query optimizer during operations.

Validation of conceptual and internal consistency (R-43) is provided by the simulators. Validation of external schemata has not been specifically addressed in RIPS. R-46 is satisfied through simulation before implementation, and by the RIPL preprocessor of the GEUF and the RIPL

compiler during operations. The requirements of R-47 are satisfied to the extent that candidate solutions to the entry of large volumes of data can be evaluated by simulation.

In addition to discrete-event simulation, RIPS has addressed the assessment of inherently unquantifiable characteristics of candidate DBMS software in the evaluation and selection process. Among these are such characteristics as vendor support, ease of use, reliability, conformance to standards, and others. The basis of this methodology is recognition that any two candidate software packages can be made functionally equivalent in virtually every respect through additional programming and/or contracted services. The cost or time required to provide this additional effort for each characteristic is the key parameter in the decision process. These extensions to DBMS simulation performance analysis capabilities provide a complete database system design and selection methodology, satisfying R-3, R-4, R-5, and R-12.

KM EXTENSIONS TO RIPS

KM concepts require capabilities that have been considered for RIPS, but the method of their implementation--or whether they will be incorporated at all--has not been determined. These include time dependencies (R-51), application of probablistic rules (R-50), context integrity (R-45), special data presentation techniques (R-32, R-34), dynamic network resource allocation (R-28), dynamic data structure change (R-11, R-24), automatic schema generation (R-23), concurrency resolution (R-14), improved security techniques (R-13), and text processing. The potential for incorporating these requirements in RIPS is discussed below.

Time Dependencies

All data relations are time-dependent; but because the majority of user's needs pertain to current data, time-dependent specifications are implemented by existing systems for only those data that the user anticipates will require frequent retrieval for specific times. The relations

EMP(E#,NAME,MARITAL-STATUS,ADDRESS,...)

SAL-HIST(E#,DATE,SAL)

recognize that an employee's salary changes, and implements the time-dependent knowledge of such changes. Of course, the employee's marital status, address, and even name are also subject to change with time, but ready access to this specific knowledge is not anticipated, and its retention is usually in the form of archived databases. Within the period of a stable information structure, say time T_1 to T_4, there may be one implementation (e.g., checkpoints, archival files) for T_1 and T_3 and another (e.g., on-line database) for T_4. A query to find an employee's address at time T_2 can be answered by applying the transaction log covering T_1 to T_2 against the T_1 archive. Thus, the transaction log represents the database change from T_1 to T_4, and time dependencies require that we view the transaction log as a natural part of the database.

The KM concept would make such queries possible, and the processing would be clear to the user. RIPS already envisions the potential for relations being implemented by multiple schemata, including the transaction log, and in this case, instances of EMP relation attributes are distributed by restrictions on the (unstored) time domain, just as tuples in SAL-HIST could conceivably be physically distributed by restrictions on the (stored) date domain.

A unified view of the time dependencies of relations would recognize that all relations have a time domain, and the single relation

EMP(E#,NAME,SAL,MARITAL-STATUS,ADDRESS,TIME)

would suffice as the information structure for any time-dependent query, including salary history, marital-status history, etc, and query language need not change to accommodate specification of time dependencies. However, this extension requires modification of current concepts of tuple identifiers, derivations, user views, and other components of RIPS, and it is not clear what the effect of such modifications might be.

Application of Probablistic Rules

Some types of probablistic rules (fuzzy logic) can easily be handled by existing RIPS concepts. In Reference 21, Figure 9 is presented as an example of fuzzy knowledge. In this work, the authors present a system, Fuzzy-Set-Theoretic Data Structure (FSTDS), for implementing such knowledge so that queries like "what does a bat belong to?" can be answered.

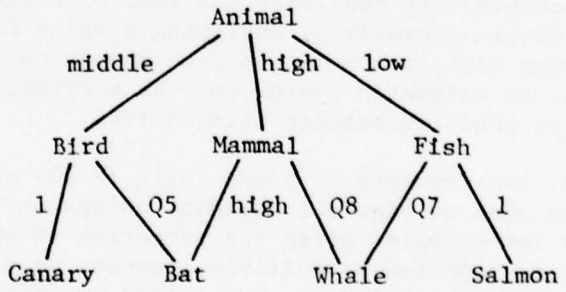


Figure 9. Example of fuzzy knowledge[21]

The edges of the graph in the figure represent a compatibility factor for the association between the nodes that has meaning to users of such knowledge. Thus, bird is associated with anaimal with a compatibility factor "middle," whatever that means.

In the RIPS information structure, semantic concepts are made explicit; thus, a relational view of the knowledge would recognize the concept of

---

21. Masaharu Mizumoto, Motohide Umano, and Kokichi Tanaka: "Implementation of a Fuzzy-Set-Theoretic Data Structure System," Presented at Third International Conference on Very Large Databases, Tokyo, Japan, October 1977.

compatibility as perhaps the relations

ANIMAL-TO-CLASS-COMPATIBILITY-ASSOCIATION

CLASS-TO-SPECIE-COMPATIBILITY-ASSOCIATION.

The attributes and instances of these relations, renamed for convenience, are shown in Figure 10. Now, the query "what class does the species bat belong to and with what compatibility?" is easily expressed in RIPL as

```
GET S .OF. CLASS-SPECIE/CLASS,COMP .WHERE. SPECIE = 'BAT'
```

which would produce the same answer as that produced for the previously stated query to the FSTDS system.

Animal - Class (Class, Comp)

Class	Comp
Bird	middle
Mammal	high
Fish	low

Class - Specie (Class, Specie, Comp)

Class	Specie	Comp
Bird	Canary	1
Bird	Bat	0.5
Mammal	Bat	high
Mammal	Whale	0.8
Fish	Whale	0.7
Fish	Salmon	1

Figure 10. Possible relational view of fuzzy knowledge.

The example demonstrates that, while fuzzy knowledge exists, in real-world situations, people must deal with the lack of certainty, or probabilities, by some means--commonly by assigning a value factor to the alternatives and making decisions based in some manner on these values. To repeat the process, an automated system must be provided with the same values and rules for choosing between alternatives.

In this regard, the concepts of fuzzy logic (R-50) and validity values (R-44) converge when we view the validity values as the semantics of uncertainty. Thus for example, given the situation in which the location of an airfield may be supplied by multiple sources, we can assign a validity value to each whose semantics are made clear to users. A value supplied by a satellite may be considered more reliable (i.e., have a higher validity value) than that supplied by an aircraft, which in turn is more reliable than that from a ground observation. By assigning numeric validity values to each type of source, we can declare the semantics in the relations

AIRFIELD (NAME, TYPE, #RUNWAYS,...)

AIRFIELD-LOC (NAME, LOC, VALIDITY)

A query to find the most probable location of airfield X can be stated in RIPL as

```
GET S .OF. AIRFIELD/LOC,VALIDITY .WHERE. NAME='X'
GET T .OF. MAX/MAX-A .WHERE. A'=S/VALIDITY'
PRINT U .OF. S/LOC .WHERE. VALIDITY=T/MAX-A
```

In RIPS, the concept of "most probable airfield location" can be specified in general through the use of derivations, and this knowledge can be made available to users. Thus, the specification

```
S(AIRFIELD)=AIRFIELD-LOC/NAME,LOC,VALIDITY .WHERE. NAME=AIRFIELD/NAME
T(AIRFIELD)=MAX/MAX-A .WHERE. A'=S(AIRFIELD)/VALIDITY'
MOST-PROB-LOC(AIRFIELD)=S(AIRFIELD)/LOC ,WHERE.
                                    VALIDITY=T(AIRFIELD)/MAX-A
```

stored in the DD/D extends the information structure, producing the derived attribute

"MOST-PROB-LOC"

in the context of airfield as

AIRFIELD(NAME, TYPE, # RUNWAYS, MOST-PROB-LOC, ...)

and now the $RIPL_n$ query can be stated by users as

```
GET S .OF. AIRFIELD/MOST-PROB-LOC .WHERE. NAME='X'
```

which the GEUF will reduce to a $RIPL_o$ query.

In keeping with KM philosophy, the knowledge concept is visible to the knowledge manager because the derivation is accessible from the DD/D. Additional explanation of the history or rationale behind the concept can be stored in the DD/D and automatically displayed to users querying the derived attribute through the RIPS concepts of predefined queries.

The preceding is not intended to be a final statement of the application of fuzzy logic, probablistic rules, or validity values, but only to describe current RIPS concepts. Further investigation of the subject will lead to a unified concept for handling uncertainty by making the semantics visible to the using community.

Context Integrity

We have been investigating the use of Dana Scott's lattice theory logic[22,23,24] for representing unknown or unavailable and inconsistent

22. Dana Scott: *The Lattice of Flow Diagrams*. Technical Memo No. PRG-3, Programming Research Group, Oxford University Computing Laboratory, 45 Banbury Rd, Oxford, England.

23. Dana Scott: "Logic and Programming Languages," *Communications of the ACM*, Vol 20, No. 9, pp 634-641, September 1977.

24. D. S. Scott: "Data Types as Lattices," *SIAM Journal on Computing*, 5, 1976.

information for knowledge management. This lattice, or four-valued logic shown in Figure 11, has in addition to the usual truth values the symbols bottom ($\perp$) and top ($\top$). Bottom means that the value is unknown or unavailable (at least to the particular user) at this time. Top means that information is inconsistent. We view the symmetry of this lattice as a useful model for, on one hand, attempting to retrieve information that is not available, and on the other, attempting to add information in violation of integrity assertions.

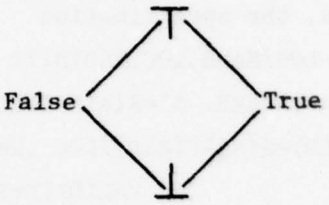


Figure 11. Scott's lattice

Scott's theory introduces an information theoretic ordering relation ($a \subseteq b$), which means that a is consistent with b as far as it goes, but that b may have more information. Thus, $\perp \subseteq$ false, $\perp \subseteq$ true, false $\subseteq \top$, and true $\subseteq \top$. This ordering can be extended to all types of data as well as truth values. Figure 12 shows the lattice applied to numbers. In this case, bottom represents an unknown or inaccessible number, and top represents an attempt to assign two different numbers to a data element that may have only a single value.

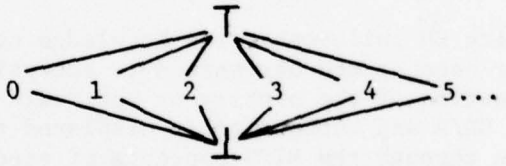


Figure 12. Scott's lattice applied to numbers.

Based on the ordering, the theory defines limits of sequences and continuity in a manner similar to mathematical analysis or topology. The notation of a monotonic sequence approaching a limiting value is used to mean successive increments of consistent information approaching the maximum truthful information available.

The objective of this research is to combine the concepts of integrity, authorization, and concurrency resolution into a single theory and implementation.

Dynamic Network Resource Allocation

The issue of resource requirements (R-28) is addressed in RIPS through the database system simulator to the extent that, for a given application

and DBMS, the host system, traffic demands, and distributions required by the application can be analyzed for each data user, source, channel, and storage device defined. These parameters are used in determining the adequacy of the given system to accommodate the application. Thus, while no automated network resource design or dynamic resource allocation capability is being pursued, some key elements in any such decision-making model are included in the RIPS QDD and measurable by the simulator.

Dynamic Restructuring

The requirement for dynamic restructuring (R-11, R-24) implies the need for data-use descriptions that constitute the decision-making criteria for selecting from among alternative data structures. In RIPS, the QDD serves this purpose but is used for data structure evaluation as opposed to data structure design. However, the following example demonstrates the potential for employing the QDD and other RIPS components in a cybernetic system.

The QDD parameter, associated qualification rate (AQR), describes the rate at which an attribute is used in qualifying tuples of its associated relation. Consider the relation

EMP(E#,NAME,SECURITY-CLEARANCE, ...)

and the environment in which queries about employees are commonly qualified by particular security clearances. If the rate of such qualification is high enough, the DBA could choose to implement an index of security clearance values to facilitate quick retrievals. If, over a period of time, the organization becomes less involved in security projects, such qualifications may become extremely rare. Now the index may be consuming resources disproportionate to its utility. If the QDD parameters were maintained dynamically by sampling user's production queries, the value of AQR for security clearance would eventually fall below some DBA-prescribed threshold, indicating that there is no longer justification for the index. In RIPS, it would be relatively simple to define this situation as a trigger to execute partially predefined queries that would change the DIAM descriptions of the implementation in the DD/D and to issue DDL statements to the DBMS to eliminate the index.

While the preceding is a rather simple example of dynamic restructuring, extension to more complex considerations appears promising. However, serious problems can result from treating changing profiles as permanent conditions, when in fact they are anomolies. Some intervention may always be required to determine the underlying reason for measured changes and further actions initiated on the basis of these findings.

Automated Schema Generation

Many of the considerations discussed in the preceding paragraph pertain here. Automated generation of the conceptual schema would require some statement of the functional dependencies that exist in the real

world. Excellent work has been done in this area (e.g., Reference 12) in which the result of a collection of functional dependencies is a set of third-normal-form relations. However, because the solution is not unique, the utility of these techniques in a production environment is questionable. We are investigating a fourth normalization that will include further real-world constraints and lead to a unique solution.

Except for default display formats, external schemata generation can be automated only to the extent that the interfacing techniques provided for the user can be made easy to use. This is because display formats are generally dictated by external considerations (e.g., government forms, industry standards, etc). Providing a suitable user interface is already envisioned in RIPS, but the details are properly the subject of human factors. A good example of such an interface is Query by Example and the System for Business Automation,[14] which can be user defined in RIPS using current concepts.

Internal schemata generation can be automated by modeling a designer's decision-making process within the alternatives offered by his DBMS. The example in the preceding paragraph illustrates this concept using the QDD parameters of the application in the decision-making process.

Concurrency Resolution

In RIPS, the problem of concurrency resolution is considered to be partially resolvable at the information level. In general, however, if multiple sources can update the same data, so that one update may supersede another, there is some question of organizational consistency. The example often presented involves a case in which one source wants to give a specific employee a 10% increase in salary and another source wants to increase the salary of all employees by 5%. The order in which these queries are processed will affect the final salary of the specific employee. However, the problem is not one of data processing but one of organizational policy. If such an eventuality can occur in some context, then rules for which query is to be processed first must be provided.

In many cases, the analysis required to derive such concurrency rules will result in recognition that the semantics of the attribute subject to update is such that there are in fact multiple attributes in question. For example, if the location of an airfield can be provided by two sources, retention of both may be useful and must therefore be recognized in the information structure. Thus, rather than the relation

AIRFIELD(NAME,LOC, ...)

we have

AIRFIELD(NAME,LOC-PER-SOURCE-1,LOC-PER-SOURCE-2, ...)

---

12. Morton M. Astrahan and Donald D. Chamberlain: *Implementation of a Structured English Query Language.* RJ1464, IBM Research Center, San Jose, California, October 28, 1974.

14. M. M. Zloof: "Query by Example," *Proc. National Computer Conference*, AFIPS Press, Vol 44, 1975, pp 431-438.

and the order in which two simultaneous updates are processed is immaterial with respect to the final values. The example can easily be expanded to include recognition of probablistic or accuracy values owing to the source.

In general, the case of retrievals for data currently being updated can be handled at the information level by examining queries in process before submitting the retrieval. If the same relations are involved in the update as in the retrieval, the retrieval must be held until completion of the update. In a distributed environment, the knowledge that a pertinent relation is being updated by a remote site may not be available to the site where the retrieval query originated. A satisfactory solution to this problems remains to be found.

In an environment in which time dependencies may be expressed in the query language (see Time Dependencies), the concurrency resolution problem is expanded for the previous example. If a retrieval query is received specifying time T_1, and an update has been processed at time T_2, the update transaction must be backed out in order to answer the query. In general, however, the concurrency resolution routine can determine whether any conflicting updates have been processed if it has access to the transaction log. If no updates have occurred between the time specified in the retrieval query and the current time, the query can be processed immediately.

Improved Security Techniques

Concepts of ensuring security or authorization at the information level have been described. Whether or not additional security techniques (e.g., hardware-provided encryption, etc) will affect the concepts remains to be determined.

Data Presentation

RIPS envisions commonly used but diverse data presentation techniques including graphics, but has not addressed such concepts as imposing digitally generated data over externally provided formats, holographic imagery, etc. RIPS' primary concern is to provide the internal representation to a device driver that will result in the externally visual representation sought. The major apparent problem is that users think in terms of visual representation and want to describe displays in similar terms. The GEUF is intended to provide this interface to facilitate declaration of formats. The corresponding internal representations appear to be describable in DIAM terms, at least for commonly used devices and displays, as described later, but the choice of display formats for particular applications and users is the subject of human factors and human engineering, limited only by available devices.

RIPS has proposed the extension of DIAM descriptions to include specification of two-dimensional displacements, simply because displays are viewed in two dimensions. It remains to be determined whether this extension will suffice for the types of data presentation envisioned by KM.

Text Processing

Conceptually, a text processing system is a database application in which relations that comprise the information structure or conceptual view are documents, pages, paragraphs, sentences, words, etc. Database functions are required to retrieve and maintain text data in the same way as for any other entity representations, but of course some storage structures are much more efficient for text processing than other types of data processing.

In general, characteristics that cloud these similarities are the user's language, limited information structure, uniform storage techniques, and document-oriented display formats. Also, text processing systems make extensive use of temporary (working) storage where documents and pages are modified in a fast-access temporary storage, then placed in a permanent file on completion.

Conceptually, RIPS accommodates all these characteristics, including the declaration of temporary relations that could well be documents or pages, and thus envisions text processing as a natural part of the MIS. However, the precise relations that constitute a sufficient information structure for generalized text processing and application of the RIPS concepts in defining a suitable user's language, display formats, and storage structures have not been analyzed in this specialized application.

RIPS EXTENSIONS TO KM

The major conceptual extension of KM that RIPS provides lies in the degree and methodology of providing visibility to knowledge concepts. In the RIPS view, management of knowledge concepts is precisely the management of application functions, and for effective control, the functions must be visible--not bound indistinguishably in application programs. This major concept has led RIPS to eliminate, as much as possible, the common practice of application program development, thus allowing users to specify 'what' information is required, not 'how' it is to be obtained. The potential of this concept is a solution to many problems of applications programming practice, including modularization, structured techniques, language usage, programming standards, etc.

KM concepts require a comprehensive set of capabilities--far more than any existing system provides. However, even if all the requirements were to be provided by a production system, implementation methods could be so diverse that the total system would be unmanageable. For example, if schemata descriptions for a production system differ from those for a simulation system, interfacing problems could result in such a lengthy definition phase that timely results would be impossible. It is not only a requirement that KM capabilities be provided, but it is equally a requirement that techniques employed be consistent to be manageable.

RIPS provides this compatibility throughout. The same QDDs are used for query optimization as are used for simulation; schema descriptions use the same formalism for internal and external schemata and for internal schemata of both the production system and the simulator; the DD/D declarations and retrievals use the same language as productions queries; etc. Thus, RIPS concepts extend the requirements to include those of consistency in implementation.

The RIPS concept of providing independence of queries from their implementation extends to external or display formats. As previously stated, this independence is necessary to relaize the ANSI-X3-SPARC architecture. Thus, while KM requires orderly migration of internal and external functions, RIPS makes such requirements more explicit by defining the elements for realizing them.

The KM concept envisions a high degree of knowledge of the requirements for nodes in the network, especially in processing optimization and performance analysis. In RIPS, these requirements are made explicit through quantitative data decriptions that are employed as a natural element of system architecture. The effect of this concept is a unified view of processing requirements and information management--both use the same formalisms.

FUNCTIONAL ALLOCATION OF KM REQUIREMENTS TO RIPS

This section summarizes the functional capabilities of RIPS and shows the correlation to KM functions. Table 1 contains the allocation assignments. Each column of the matrix represents a functional component of RIPS and is described below. The three columns at the right of the table indicate that the conceptual foundation required to accommodate corresponding KM functions is incomplete, undetermined, or not considered part of RIPS. Where entries are made in one of these columns and in one or more of the other columns, the corresponding KM requirement is only partially satisfied by the current RIPS concept as further described below.

Unless otherwise noted, KM requirements reference those listed under Synopsis of KM Requirements (page 28ff).

Generalization of Requirements

Generalization of requirements recognizes the fact that requirements are not only necessary for an initial system implementation, but because they are evolutionary, they are also the basis for changes to implementations. The original statement of requirements should be maintained to allow visibility of gradual changes that foretell the need for corresponding implementation changes to accommodate them.

To be sufficient, virtually every change to an implementation and every choice made between alternative techniques should be justifiable

Table 1. Allocation of KM functional requirements to RIPS

*		Requirements Generalization	S/W Evaluation Methodology	Performance Evaluation	Processing Generalization	Context Generalization	Format Generalization	Integrity Generalization	Authority Generalization	Concurrency Resolution	Generalized Data Accessing	Specification Generalization	Concepts Not Analyzed†	No Formal Concept†	Outside RIPS†
1	KM Facility					X						X			
2	Organizational Model	X													
3	Software Selection Guidelines		X												
4	System Evaluation Procedures		X	X											
5	Experiment Design													1	
6	Standardized End-User Techniques				X	X	X								
7	Distributed Database Descriptions	X									X	X	1		
8	Data Dictionary/Directory											X			
9	Metadata Management							X	X						
10	Performance Analysis Tools			X											
11	Dynamic Data Restructuring													2	
12	Access Method Evaluation	X	X												
13	Improved Security Techniques								X						1
14	Deadlock Resolution									X					2
15	Database Recovery														3
16	Database Restoration														4
17	Semantic Consistency					X		X							
18	Logical-to-Physical Mapping		X								X				
19	Performance Predictions			X											
20	Performance Parameters	X	X												
21	Schema Descriptions										X	X			
22	Database Migration										X				
23	Automated Schema Design													2	
24	Schema Descriptions										X	X		2	
25	Internal Schema Validation	X	X	X											
26	Application Migration				X	X	X				X				
27	Database Synchronization										X				
28	Dynamic Resource Allocation													3	
29	Subnetwork Models										X	X			
30	Local & Global Database Views					X					X			4	

Table 1. (concl)

*		Requirements Generalization	S/W Evaluation Methodology	Performance Evaluation	Processing Generalization	Context Generalization	Format Generalization	Integrity Generalization	Authority Generalization	Concurrency Resolution	Generalized Data Accessing	Specification Generalization	Concepts Not Analyzed†	No Formal Concept†	Outside RIPS†
31	User Interface Migration						X				X				
32	Data Presentation Techniques														1
33	Alert Processing				X	X									
34	Data Display Techniques						X								1
35	Database Browsing				X	X					X	X			
36	Computer-Aided Instruction					X									
37	Query Processing Optimization										X				
38	Access Code Generation										X				
39	Query Analysis & Simulation		X								X				
40	Computer-Aided Instruction					X									
41	Integrity Controls							X							
42	Algorithm Validation													5	
43	Implementation Validation		X												
44	Data Validity				X	X		X							
45	Context Integrity							X					2		
46	Query Validation							X			X				
47	High-Volume Data Entry						X				X			2	
48	Standard Data Entry Techniques				X	X	X								
49	Conceptual Model											X			
50	Application of Probablistics				X								3		
51	Time Dependencies					X					X		4		
52	Text Processing				X	X	X				X		5		

* Items generally refer to numbers in Synopsis of Requirements (page 28ff).

† See explanation of numbered notes on pages 59 and 60.

in terms of the requirements. Thus, they are necessary for all performance analyses and all management functions dealing with organizational data flows and real-world dynamics.

Generalization of requirements in RIPS refers to the information structure and QDD stored in the DD/D, maintained through the use of predefined queries, and accessible through the RIPL and GEUF capabilities.

DMS Software Evaluation Methodology

DMS software evaluation methodology is a set of techniques that provides a means of choosing between alternative software packages for a known application in a given environment that are both consistent and repeatable (i.e., two analysts must arrive at the same conclusion). The evaluation includes both qualitative and quantitative characteristics.

Software evaluation methodology in RIPS refers to the use of QDD, math-model and real-time simulators, and procedures for evaluating qualitative characteristics in terms of the time and cost required to provide the same qualitative characteristics for each alternative.

Performance Evaluation

Performance evaluation is a means of predicting the performance of a DBMS implementation for a known application before implementation. The results of such simulations are the basis for choosing between alternative implementations.

Performance simulation in RIPS refers to the use of the QDD and the math-model and real-time simulators.

Generalization of Processing

Generalization of processing is a means of relieving users of the need to specify how and where data in the network are accessed or derived, allowing specification of only what data are required. In RIPS, this capability is provided by:

1) Viewing algorithms as relations of the information structure in the same way that stored data are described;
2) RIPL language;
3) Processing both algorithms and data by procedures automatically generated by the QC/T regardless of their implementation;
4) Automatic reduction of $RIPL_n$ queries to $RIPL_o$ by the GEUF.

Generalization of Context

Generalization of context provides the means of predefining the parts of queries that are known a priori, allowing users to supply only the

particulars of current interest at execution time. This precludes users from having to restate entire queries each time they are required. The capabilities extend to the extremes where, on the one hand, no parts are known a priori (i.e., ad hoc queries), and on the other, all parts are known including when they are to be executed (i.e., real-time queries). Allowing user-defined specifications of when queries are to be executed, by specification of trigger conditions as the stimulus, provides alert processing.

In RIPS, these capabilities are provided by:

1) Allowing partially predefined queries to be defined in RIPL and stored in the DD/D as relations that are part of the database;

2) Mapping user-supplied data to particularize a query by updating the stored relations (in 1) through procedures automatically generated by the QC/T as directed by the GEUF;

3) Allowing stimuli to be user-defined along with specification of what predefined queries are to be executed upon receipt of a stimulus;

4) Monitoring of stimuli by the GEUF and automatic execution of the corresponding query by the QC/T;

5) Use of user-supplied user views to establish a tailored view of the database in the vernacular of a specific discipline;

6) Specification of derivations to make knowledge concepts visible by declaring their semantics, thereby extending the information structure and making such knowledge available to users in a consistent, uniform manner;

7) Use of $RIPL_n$ to reference user views and derivations, and its automatic reduction to $RIPL_o$ by the GEUF.

Generalization of Formats

Generalization of formats is a means of relieving users of specifying how display formats are to be generated, allowing specification only of what formats are required. The capability provides device independence and, in conjuction with generalization of context, allows complete flexibility in designing user interfaces.

In RIPS, this capability is provided by:

1) Specification of formats in DIAM terms, via RIPL, stored in the DD/D;

2) Use of partially predefined queries;

3) Processing user-supplied data and display formats through procedures automatically generated by the QC/T as directed by the GEUF.

Generalization of Integrity

Generalization of integrity is a means of relieving users of having to ensure that all data stored and displayed are current each time a query is executed. The degree to which integrity can be thus controlled is user-defined by assertions expressed in RIPL for each relation/attribute in the information structure.

In RIPL, these capabilities are provided by:

1) Specification of integrity assertions expressed in RIPL and stored in the DD/D, including assertions over the relations of the DD/D itself;

2) Query modification automatically appending integrity assertions to all user's queries by the GEUF;

3) Evaluation of all query predicates by the QC/T before execution, thereby processing only valid queries.

Generalization of Authorization

Generalization of authorization provides a means of controlling access to stored and derived data by specifying what data are prohibited to specific users or sources, in terms of the information structure.

In RIPL, these capabilities are provided by:

1) Specification of what authorization constraints are to be imposed via RIPL and stored in the DD/D, including relations in the DD/D itself;

2) Query modification by the GEUF, automatically appending authorization constraints to all user's queries;

3) Evaluation of all query predicates by the QC/T before execution, thereby processing only authorized queries.

Generalization of Data Accessing

Generalization of data accessing recognizes that there is a consistent model for the description of alternative implementations for a given information structure with respect to stored data, and that exploiting such descriptions allows mapping of representation-independent queries into representation-dependent queries. Coupled with the syntactic translator, representation-dependent queries can be expressed in the language of diverse data management systems.

In RIPS, these capabilities are provided by:

1) Allowing descriptions of distributed implementations in terms of the DIAM model and the information structures, and storing them in the DD/D;

2) Reducing $RIPL_o$ queries to representation-dependent language (RDL) by the QC/T;

3) Translating RDL into a DMS language by the QC/T;

4) Translating data returned by DMSs into canonic representation of the information structure defined by RIPL queries by the QC/T.

Generalization of Specifications

Generalization of specifications recognizes the fact that, for data processing functions to be manageable, specification of such functions must be available to managers in a consistent manner.

In RIPS, this capability is provided by:

1) Viewing all functional specifications as relations stored in the DD/D;

2) Including these relations in the information structure, thus making them available just as any other data in the database;

3) Allowing specifications of integrity, authorization, user views, etc over the relations that contain the specifications;

4) Allowing all capabilities of RIPS to be applied to specifications.

Proposed Concepts Not Analayzed

Entries in this column of Table 1 correspond to the following notes:

1) Automatic update of QDD parameters is only partially provided. Populations and population distributions of entity representations can be periodically determined by issuing applicable count queries to each node in the network. However, the dynamics can only be maintained current for remote nodes in cases where a RIPS package is installed. For nodes without a RIPS package, specialized applications for this purpose must be implemented.

2) Generalization of integrity primarily addresses the case of updates. The use of four-valued logic in query processing addresses retrievals including derivations (refer to Context Integrity).

3) While user-defined probabilities are considered in generalization of context, fuzzy logic and inferencing have not been fully analyzed (refer to Application of Probablistic Rules).

4) Time dependencies have not been analyzed, but the inclusion of time domains in relations appears promising (refer to Time Dependencies).

5) Text processing techniques have been proposed but not fully analyzed (refer to Text Processing).

No Formal Concept Available

Entries in this column correspond to the following notes:

1) The design of experiments is now largely intuitive. Formal techniques are needed but require further research. No proposed solution is available.

2) Data restructuring is provided when both the source schema and target schema are predetermined, provided through the use of predefined queries whose output format is the target schema. Automatic generation of an internal target schema that is optimized in some sense appears feasible when the target schema is selected from some small set of alternatives, and the rules for choosing between them are known. Automatic generation of a conceptual schema has been proposed[12] but not analyzed for RIPS. No formal concepts are available for automatic generation of external schemata, although default schemata are provided.

3) Concepts for dynamic resource allocation have not been proposed.

4) The subject of parallel processing is accommodated in the sense that subqueries to different nodes have the potential for simultaneous processing accommodated by the QC/T. Parallel processors at a single node have not been addressed.

5) Algorithm validation has not yet been addressed. Extensive work has been accomplished in this area, including formal proofs, but no formal concept has been selected for application to RIPS.

Requirements Outside RIPS

Requirements outside RIPS include:

1) All external representations include considerations that are the subject of human factors and engineering that are outside the scope of RIPS.

2) The QC/T is not intended to be a DBMS as such. We have recognized
3) that a DBMS built on the DIAM theory would be useful and could be
4) employed recursively for both external and internal data management. In such an environment, techniques for (2) deadlock resolution and locking schemes, (3) database recovery, and (4) restoration and restructuring would have to be provided for any current implementation. However, in the environment intended, where the QC/T interfaces with existing DBMs, these techniques are considered the responsibility of that DBMS and therefore outside the scope of RIPS. Exceptions are discussed under Factual Knowledge Subsystem (FKS).

---

12. Morton M. Astrahan and Donald D. Chamberlain: *Implementation of a Structured English Query Language*. J1464 IBM Research Center, San Jose, California, October 28, 1974.

CURRENT STATUS OF RIPS

Information Structure

A baseline information structure is proposed as described under Information Structure (page 11). Four activities may affect final specifications:

1) A complete analysis of the description of algorithms in relational terms. The major problem to be considered is representation of algorithms that appear to require ordering (e.g., matrix operations) or ranking (e.g., statistical analyses). Concepts have been proposed.

2) Time dependencies;

3) Fourth normalization;

4) Verification of $RIPL_o$.

Representation-Independent Programming Language

Existing concepts are undergoing proof and completion demonstrations. Paper is in progress. Work remaining includes:

1) Final information structure (includes time dependencies);

2) Proof and completion analyses;

3) Query balancing analysis;

4) Final syntax.

Data Dictionary/Directory

The minimum set of relations that will be required cannot be determined until all other specification descriptions are. Determination of which attributes are to be under system (preprogrammed) authorization control remain to be determined. Basic concepts of access and maintenance are proposed but require completion of the GEUF and QC/T for validation. Although access to the DD/D is generalized, some sample set of user-defined DDL (see Data Description Language) must be provided for RIPS prototype demonstration and will be accomplished during test application description and encoding.

QDD needs to be re-evaluated in the KM environment and be cast in relational terminology. Predefined query specifications require completion of RIPL. Stimulus specifications require further analysis, including update of QDD. Some stimulus specifications have been proposed. Display formats described in DIAM terms appear to require an extension to DIAM to describe two-dimensional displacements. This analysis has not begun, but will include analysis of text processing requirements. Specifications of orderings, symbols, and graphics have not begun.

Integrity assertions and authorization constraints have received considerable attention by other workers, which appears sound. However, we

have not yet attempted to incorporate them in RIPS. Both require completion of $RIPL_o$.

Basic concepts of derivations have been investigated, and their requirements are well understood. Final descriptions require completion of $RIPL_o$ and are required for completion of $RIPL_n$. The same pertains to user views.

DIAM descriptions of internal implementations appear to be complete with some recent extensions, but these remain to be validated. This work is in progress.

Generalized End-User Facility

Completion of the GEUF requires completion of all application function specifications. Major components to be developed are:

1) Query modification to append integrity and authorization constraints to user queries;
2) Query preprocessor to reduce $RIPL_n$ queries, which are in terms of user views and derivations, to $RIPL_o$ queries, which are in terms of the basic information structure;
3) Stimulus monitor to direct the QC/T to execute queries corresponding to both user-supplied and internally generated stimuli.

No programming has begun on the GEUF.

Query Compiler/Translator

Preliminary query decomposition is complete using currently defined $RIPL_o$. Final validation has not begun. The search-path enumeration algorithm is complete for a significant subset of access path descriptions and is being validated. Restrictions that specify a specific value and restrictions based on value ranges (e.g., employees whose salary is greater than X) have not been implemented. Considerable work has been done on search-path selection criteria based on cardinality of search paths calculated from QDD population descriptions and implemented string path. Some calculations (e.g., relations having multiple attributes as their identifier) have not been solved. Only uniform distibutions are implemented, although the description of complex distributions (i.e., normal, Zipfian, empirical) are implemented in QDD. No heuristics have been implemented. A baseline RDAL has been proposed and is partially implemented. No effort has begun on syntactic translation.

All programming has been accomplished using the math model simulator as a test bed for validation of concepts. Consequently, no data are returned for current test queries; and compilation of data for display is conceptual.

Math-Model and Real-Time Simulators

As extensions to basic concepts are developed, they are being incorporated in the math-model simulator to provide a test bed for evaluating QC/T concepts and other descriptions. By continuing this process, the math-model simulator is being maintained to current concepts along with their development. The result is that, at completion of the QC/T prototype development, the math-model simulator will be relatively current. However, some reprogramming will be required for operating efficiency because the modifications will be ad hoc. These improvements will also have to be incorporated in the real-time simulator. For the most part, algorithms developed for the QC/T will be used intact for the simulators.

DMS Software Evaluation Methodology

Software selection methodology is complete but not documented. Because the methodology includes use of the simulators, they must also be complete.

RIPS DEVELOPMENT WORK PLAN

This section discusses a RIPS development plan and facility requirements to support the development. In addition, some estimates of performance in an operational environment are provided, projected from empirical results of current research software.

Development Plan

Figure 13 is a work plan for completing the conceptual design, developing prototype software, and providing applicable documentation for a KM test bed using RIPS. The plan includes a test and checkout phase designed to validate RIPS software in an existing distributed information system on a noninterference basis. That is, the test phase is to demonstrate the technical aspect of the system--not the KM concepts with respect to management issues.

Figure 13 schedules the tasks to be performed, as discussed in the previous section, and contains rough order-of-magnitude estimates of manpower and costs of computer use for each. Computer cost estimates are projected from empirical results of current developmental work of a similar nature using Martin Marietta computing facilities, as discussed in the next section.

A total of 58 man years is estimated over a 4-year development period. Total computer support is 120 IBM computer units at an approximate cost of $200/unit or a total of approximately $24,000.

The proposed schedule, in quarter years, is phased to incorporate task results in succeeding tasks as necessary and to provide a relatively constant staffing level.

Year	1				2				3				4				Task Totals	
Quarter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	M/M	Completed Units
Time-Dependencies Analysis	2	2	2														18	
Algorithm Descriptions	1	1	1														9	
Basic Information Structure Description	1	1	1														9	
$RIPL_o$ Completion	1	1	1	1													12	
Algorithm Execution Description				1	1	1											9	
Display Format Description	1	2	2	2													21	
Derivation Description			1	1	1												9	
User View Description				1	1	1											9	
Integrity Description					1	1	1										9	
Authorization Description					1	1	1	1									12	
Concurrency Description					1	1	1										9	
Predefined Query Description					1	1	1										9	
Stimulus Description				1	1	1	1										12	
Odd Extensions				1	1	1											9	4
$RIPL_n$ Completion							1	1	1								9	
Query Modification Procedure								2	1	1							12	6
Query Preprocessing Procedure							1	1	1	1							12	6
Stimulus Monitor Procedure								1	1	1	1						12	6
Concurrency Resolution Procedure									1	1	1	1					12	6
Odd Dynamic Update Procedure									1	1	1						9	5
GEUF Completion											1	2					9	6
Query Decomposition	1	1	1	1													12	6
Path Enumeration Procedure		1	1	1	1	1											15	8
Search-Path Heuristics Description				1	1	1	2	1									18	
Search-Path Optimization Procedure						1	2	2	2								21	10
Search-Path Selection Procedure								1	1								6	3
RDAL Description			1	1	1												9	
RDAL Generation							1	1	2	2							18	6
Syntactic Translation								2	2	2	2						24	6
Data Compilation									1	2	2						15	6
Communications Interface									1	1	1						9	5
Device-Driver Description					1	1	2										12	
DD/D Completion												2					6	3
QC/T Completion											1	2					9	6

Figure 13. RIPS development plan.

Year	1				2				3				4				Task Totals	
Quarter	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	M/M	Completed Units
QC/T-GEUF Completion													2	2			12	6
Test Application Description												1	2	3			18	
Test Application Encoding													2	1			9	3
Test Application Execution & Evaluation														1	2	3	18	6
Math-Model Simulator Enhancement												1	1	2	2		18	3
Real-Time Simulator Enhancement														2	3		15	3
DMS Software Selection Methods															1	2	9	1
Subtotal	7	9	11	12	12	12	12	12	12	12	11	11	11	11	8	5	504	120
Technical Management	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	96	
Technical Support - Document & Edit Services	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	48	
Project Management	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	48	
Project Support																		
Total Manloading	11	13	15	16	16	16	16	16	16	16	15	15	15	15	12	9		
Man-Months (M/M)	33	39	45	48	48	48	48	48	48	48	45	45	45	45	36	27	696	

Figure 13. (concl).

The following are specifically excluded for the reasons indicated:

1) Dynamic data restructuring. No formal theory for the general case is known, and the advisability of total automation is questionable.

2) Dynamic resource allocation. Same as 1 above.

3) Simulation experiment design methodology. As pointed out earlier, experiment design is largely intuitive. No formal theory is known for the environment envisioned, although statistical methods of adequate sample size, sample selection, etc are well developed for other disciplines and should be applicable.

4) Automated schema design for internal and external schemata. For conceptual schema design, the formalisms will be developed, but their automation is excluded. No formal theory is known for internal or external schemata.

Facility Requirements

Existing software (math-model simulator and developmental query compiler) is programmed in ANSI FORTRAN and is operating on Martin Marietta's IBM370-168 VS with TSO. Earlier versions of the MMS were run on CDC 6500 and Univac 1108.

The executable portion of the program requires approximately 150K bytes (IBM). However, because the data of the application under study (i.e., Information Structure, QDD, DIAM descriptions, etc) are implemented as program data arrays, the current version, configured for a large application description, requires approximately 4M bytes. While this size presents no problem in the IBM Virtual System, the current version would require reprogramming (primarily for developing a DMS to manage the data arrays in disk storage) to run on CDC or Univac due to core size limitations.

Development to date has been implemented in a research environment directed primarily toward verification of concepts. Consequently, no attempts have been made to optimize the program, either for performance or size.

Estimated Performance Characteristics

The following predicted performance of the QC/T process is projected from existing software execution. A sample application has been designed to evaluate and demonstrate the concepts as they are being developed. The information structure is shown below over which typical Database Task Group (DBTG) access models for potentially distributed portions of the IS have been described in DIAM descriptions. Note that the DISTANCE relation could represent either stored data or an algorithm.

PLANES(ID,TYPE,MAX-SPEED,RANGE,UTM-GRID-NO,TIME-RPTD,HEADING)

ATLAS(GEO-POLITICAL-NAME, UTM-GRID-NOS)

SENSORS(TYPE,WEIGHT,FREQUENCY,PULSEWIDTH)

PLATFORMS(ID-NUMBER,PLANE-ID,SENSOR-TYPE,DATE-INSTALLED)

DISTANCE(UTM-1,UTM-2,KILOMETERS)

Nine typical queries are currently described, the most complex being as follows:

"Print the ID, max-speed and heading of all planes that have sensors with frequency of 12 and sensitivity of either .1 or .2, or, frequency of 12 and pulsewidth of 3, or, pulsewidth of 3 and sensitivity of either .1 or .2, that are located over Crete."

In RIPL, this query would appear as:

```
GET S1 .OF. PLANES/ID, MAX-SPEED,HEADING .WHERE.
      ID=S2/PLANE-ID .AND. UTM-GRID-NO= S3/UTM-GRID-NOS
GET S2 .OF. PLATFORMS/PLANE-ID .WHERE. SENSOR-TYPE=S4/TYPE
GET S3 .OF. ATLAS/UTM-GRID-NOS .WHERE. GEOPOLITICAL-NAME='CRETE'
GET S4 .OF. SENSORS/TYPE .WHERE. FRQUENCY='12'
      .AND. SENSITIVITY=('.1','.2') .OR.
      FREQUENCY='12" .AND. PULSEWIDTH='3' .OR.
      PULSEWIDTH='3' .AND. SENSITIVITY=('.3','.2')
```

Because the MMS simulator is being used for analysis, the queries are stated in RIAL, the simulator's language, which differs from RIPL in that attribute values are replaced with a value that indicates how many instances are in the qualifier. Thus, the predicate GEOPOLITICAL-NAME= 'CRETE' in RIPL appears as GEOPOLITICAL-NAME=1 in RIAL, and SENSITIVITY= ('.1','.2') appears as SENSITIVITY=2.

A QDD is defined over the Information Structure including populations, and population distributions that are defaulted to uniform distribution for the analysis. In processing this query, the search-path selection algorithm enumerated 249 different access paths, which resulted in 368 alternative access programs to cover the query. The CPU time (excluding printout) to compile the programs and compute the cardinality of access paths traversed for each, to permit selection of the optimum path, was 1.9 CPU seconds. The extension to include nonuniform population distributions in the computations is expected to be negligible. Because of the decomposition routines used, determination of which subqueries relate to which nodes in a distributed environment is also negligible.

Generation of representation-dependent access language (RDAL) programs for each of the covers will add 50% CPU time to the total, increasing the processing of the sample query to 2.8 seconds.

Addition of heuristics to extend selection optimization is expected to increase processing time but be partially offset by restricting the

number of covers (from total enumeration) to be generated. This includes analysis of how the search paths are implemented (encodings), which is not currently considered.

Syntactic translation is not implemented, but is estimated to be on the order of 0.1 second for the example. GEUF considerations are not implemented but are estimated to be on the order of 0.2 second, using default display formats.

The sample query, which is relatively complex, is estimated to require approximately 3.1 seconds of CPU time with current implementation techniques, and can probably be reduced considerably with improved techniques. A simple query--over a single relation--should require less than 0.5 second, based on similar projections.

The preceding projections are for the case of an ad hoc query. Of course, in a stable operational environment, most user's queries are either totally or partially predefined. For totally predefined queries, the potential obviously exists for saving the compiled and translated program, reducing the time to a single retrieval. For partially predefined queries, there is some potential for saving the selected search path, reducing the time to compilation and translation.

No GEUF software exists. Because it is conceptually a simple table-driven program, its projected size is estimated to require on the order of 15K to 25K bytes (IBM 370-165)--most of the program size being the preprocessor and stimulus monitor routines, and in-core buffers. Buffer size is of course an installation-peculiar characteristic and could vary greatly, depending on the number of users and the nature of their queries.

When DD/D implementation is complete, to allow application and implementation descriptions to be stored on external storage media (with proper buffering for efficiency), the total size of a RIPS package is estimated to be less than 200K bytes of storages (IBM 370-168).

No estimates of elapsed time for processing queries in a distributed environment have been attempted but, in the ensemble, the overhead required by the RIPS should be considerably offset by the optimization of search paths, which would be a difficult programming task for any other technique used. Actual processing time at remote nodes in the network is of course outside the control of RIPS, except that we are assured that an efficient program is submitted by RIPS. The time required for ad hoc or new queries in the RIPS environment would be less by orders of magnitude because the alternative is to design and write corresponding application programs for each node to be accessed, which could take days or even weeks.

Other Considerations

Current development has been accomplished by a very small permanent staff. To ensure consistency through the conceptual development, it is critical that the cadre be maintained through the first two years of development.

Timely addition of new personnel with specific skills is also critical to the work plan and is somewhat problematical. To help alleviate the education problem in view of the advanced technology being employed, we have begun negotiations with the University of Colorado, Computer Science Department, to teach a two-semester graduate course introducing the major concepts.

Estimates and plans for technology transfer and training of using organizations are not included.

CONCLUSIONS

Before presenting our conclusions, we summarize KM requirements in six major functions and compare the methods by which these functions can be satisfied by current programming techniques with the methods proposed by RIPS. This summary discussion provides a framework for analyzing the problems inherent in today's practice and determining the potential of alternative reported approaches, currently in research and development, toward a solution.

Summary of Requirements

Most KM requirements can be summarized in six major functions. The first is knowledge sharing. The advent of Generalized Database Management Systems has provided a means for managers to implement the 'data as a resource' policy. The 'knowledge as a resource' policy goes further, requiring that derivations of information from stored data and algorithms also be shared. To some degree, the current practice of application program development can make knowledge concepts available through careful partitioning of requirements and modularization of programs and subroutines. However, current practice includes allocating like requirements to larger modules, binding individual concepts to the current context and semantics. Thus, when the organization's information requirements change, it is difficult to extract required knowledge from existing applications for use in the new context. The difficulty arises partially from the lack of visibility of the concept, usually being available only in program documentation as a narrative description of the original purpose and implementation. Even when we can recognize the block of code that implements the concept, we must program a unique linkage to use the code in the new context, or repeat it in a new application program.

The RIPS solution recognizes knowledge concepts as derivations over the relations that describe stored data, algorithms, and other derivations, The concepts, either in the context of some stored data or independent of data, thus become represented by additional relations or attributes of existing relations, and are thereafter available in any RIPL queries. The concepts are visible and users need not program unique linkages for each use.

The second major function is metadata management. The KM concept recognizes that what may be specifications to an application is important information to managers, and must therefore be accessible just as any other information. In the current practice of developing application programs, requirements of the application, including the user's profiles, and implementation details are available only in the program documentation in narrative descriptions. The KM concept recognizes that user profiles constitute the organizational data flow, which is valuable information to the EA; that implementation details are necessary information to the DBAs; and that the derivations and productions are knowledge that is subject to sharing.

The RIPS solution is to include the requirements, implementation details, and derivations in the DD/D, and to include the contents of the DD/D as relations in the information structure. Thus, the DD/D becomes a natural part of the database, and access to its information is available with all the capabilities provided for other data sources.

The third major function is information integrity. The KM concept recognizes that information resources of an organization must be protected just as physical resources are. The entity name or value of individual data items being added or changed in a database is not independent of other data. While existing DBMSs can constrain individual values to a range, to specify the dependence on other data requires that application programs be developed to ensure the integrity in view of the dependence. Similarly, access to an individual data item cannot always be restricted in itself, but the restriction often depends on other associations. Again, application programs must be developed to constrain access to authorized users only. In developing application programs, integrity requirements play an important role in the partitioning and allocation of functions to program modules. Because resulting modules include both knowledge concepts and their integrity and authorization constraints, any change to either may require the module to be divided into new modules in which the knowledge concept and its constraints are properly aligned, along with the corresponding linkage to other modules.

In RIPS, the integrity and authorization constraints are defined at the information structure level, and any derivations over constrained data or processes are automatically constrained accordingly, processed by the GEUF. If the derivation changes to include more or less of the existing information structure, applicable constraints for the corresponding associations are automatically applied. Similarly, if constraints change, existing derivations are automatically constrained accordingly. In neither case are the user's queries or interfaces affected.

The fourth function is distributed access, arising from the realization that distributed information systems exist today and will continue to be required in the foreseeable future (e.g., DoD's Delegated Production Policy). Therefore, the requirement is to provide access to the distributed resources as though there were a single homogeneous implementation. Today's capabilities for accessing data in this environment

are limited to writing programs in the languages of various remote systems that contain the needed data or processes and writing another program in the language of the host node to accept the user's input and compile the retruned data in the required format.

The RIPS approach allows specification of what data or processes are required in a single implementation-independent language, and the necessary programs for accessing the remote nodes are automatically generated by the QC/T, based on DIAM descriptions of the various implementations, and corresponding syntactic translators.

Returned data are automatically compiled in the response specified by the user's query. Any special display formats or user interfaces are declared (again using RIPL) at the source node for the query and automatically processed by the GEUF and QC/T. Any changes to the distributed implementations or user interfaces are recorded in DIAM terms in the DD/D without affecting the user's query.

The fifth function is implementation flexibility. The KM concept recognizes that, in the foreseeable future, a single implementation technique cannot satisfy all performance requirements, and that existing techniques that have evolved from necessity must continue to be used. User's informations needs exist independent of the implementation. The current practice of stating user's queries by application programs that are bound to the current implementation has proved to be costly when a change is required to either the internal or external implementation.

To provide the flexibility required to allow migration of implementations to take advantage of new hardware or more efficient techniques and to react to ever changing user's needs, the information content of queries must be separated from implementation details. RIPS provides the separation through the use of RIPL to state the information needs. Details of implementations are provided by DIAM descriptions, and the combined effects of the GEUF and QC/T automatically map the queries to the internal and external representations. Thus, changes to either have no effect on the semantics of user's needs.

The sixth function is implementation aids. In view of the preceding, the choice of implementation techniques is an important consideration in the cost and performance of information systems. The statement of system requirements, including organizational data flows, data populations, etc, must be sufficient to enable designers to make rationale decisions. Because there is no formal methodology generally available that incorporates all decision parameters, and because the analyses are complex and multidimensional, current methods are largely ad hoc, developed just before their need.

RIPS recognizes the difference between issues that are quantifiable and those that are not. Quantifiable issues that include the statics and dynamics of information flow are formally described by QDD and evaluated using discrete event simulation, using the same formalisms used for the

operational QC/T. Unquantifiable issues are treated as constraints of alternative implementation techniques, including the use of commercial products, and serve to either eliminate candidates that cannot satisfy them or add in the cost to provide them.

RIPS goes further, recognizing that the requirements are not a one-time phenomenon, but are constantly changing to keep in step with real-world changes both inside and outside the control of the using organization. Thus, quantifiable requirements must be maintained, and RIPS provides the means by describing them as relations that are part of the database and allowing whatever degree of concurrency is called for in the environment.

Summary of Today's Problems

Major problems inherent in current practices of application program/DBMS development in satisfying the preceding requirements can be summarized by two characteristics. First is the time (and consequently, cost) required to automate manual systems--time during which the original requirements may change--and to implement changes in reaction to inevitable real-world changes. This time is attributable to the practice of incorporating multiple concepts in application programs, bound together by the various implementation techniques employed.

During initial development, individual functional requirements are identified, then allocated to program modules on the basis of common requirements. The commonality among individual functional requirements may include use of derivations, formats, users, data, source, response time, or other characteristics, and determination of allocations is a combinitoric problem. Missing, changing, and misunderstood requirements contribute to the problem, requiring reallocation during development, invalidating completed code, and requiring restructuring of the database, which propagates throughout the design.

During operation, a change to a single requirement cannot be made without analyzing its effects on associated encodings of other requirements that must not change.

The availability of automated information is controlled by this process. Even when the data we require are in the database, the time necessary to develop application programs to state the query determines the time in which we can retrieve the information. Once the program is developed, database implementation controls the response time. If the response time does not meet requirements, the implementation must be changed, but because the queries are bound to the implementation, those programs must also be changed--controlling the time in which the new information becomes available.

The second major problem in today's practice is manageability. Just as information is the basis for managing an organization, specifications or metadata are the basis for managing the information; and just as the

availability of information affects the management of an organization, the availability of metadata affects the management of the information.

In today's environment, we have recognized that automation can increase the availability of information and thereby improve the organization's management potential. At the same time, we have left the information necessary to manage the information to be manually implemented. The irony is that we are replacing archaic information systems with modern, sophisticated systems, but attempting to manage them by archaic information systems. As the size of information systems increases, so will the amount of metadata, magnifying information management's dilemma.

The Martin Marietta Database Research Project has concentrated on precisely these two problems, which can be reduced to a single problem of improving information availability when we include *all* information. The time required to state a query is attacked by providing a representation-independent nonprocedural programming language and the QC/T to relieve users of having to know the details of implementation. The time to obtain a response is attacked by optimizing query processing for the implementations that are provided. The time required to incorporate changes is attacked by separating application functions and generalizing their processing by the GEUF and QC/T. The availability of metadata is provided by including specifications as relations in the DD/D and including the DD/D as a natural part of the database.

Industry's Solution to Today's Problems

Industry is taking three different approaches to solving today's problems. The hardware/firmware approach is directed primarily toward providing faster response to a stated query. The major impetus is toward content-addressable memory (CAM), necessary for associative memories/processors (e.g., References 25, 26, 27, 28) and the database machine.[29] Its potential is to eliminate database design problems by providing a single method of implementation for all data. This would make the need for a single conceptual model obvious and the mapping to the data consistent, thus allowing for a representation-independent query language.

---

25. S. S. Yau and H. S. Fung: "Associative Processor Architecture--A Survey," *ACM Computing Surveys*, Vol 9, No. 1, March 1977, pp 3-28.

26. G. A. Anderson and R. Y. Kain: "A Content-Addressed Memory Design for Data Base Applications," *Proc. 1976 International Conference on Parallel Processing*, IEEE, 1976, pp 191-195.

27. D. L. Slotnick: "Logic per Track Devices," *Advances in Computers*, Vol 10, Academic Press, New York, 1970, pp 291-296.

28. C. Y. Lee and M. C. Paull: A Content Addressable Distributed Logic Memory with Applications to Information Retrieval," *Proc IEEE*, 15, June 1963, pp 924-932.

29. David K. Hsiao and Stuart E. Madnick: "Database Machine Architecture in the Context of Information Technology Evolution," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

However, requirements for knowledge sharing, metadata management, information integrity, distributed access (to existing systems). and user interface flexibility would still have to be developed to satisfy the KM concept.

The software approach includes Relational Database Systems being developed by IBM (System R)[20] and the University of California (INGRES).[13] These systems implement the conceptual data-model approach and provide a representation-independent query language. The also separate the application functions of integrity and authorization, independent of queries, and generalize their processing. Mapping of queries to the internal storage is provided by conventional DBMS techniques, with a range of implementation alternatives available to the database designer. They both require application programs to be written in a general-purpose programming language to direct the processing of derived information and external interfaces. Thus, when they become commercially available, they will not satisfy the KM requirements of knowledge sharing, metadata management, distributed access, user-interface flexibility and implementation aids.

The third approach is RIPS. It provides the functions of CAM through software via the QC/T. RIPS does not offer new computer systems technology with respect to hardware or software. What is needed exists today. Refinement of new concepts of information processing, to replace those born at the inception of computers and strengthened through years of practice, and their implementation, is the new technology offered--satisfying KM requirements.

The reported approaches and progress toward satisfying today's problems make it unlikely that a satisfactory system for realizing KM concepts will be offered in the next decade. At the current level of funding, this includes RIPS.

Conclusion

KM concepts concentrate on the use of information in government organizations, recognizing the functional and organizational requirements to permit more effective management of this essential resource. Martin Marietta's Database Research Project is concentrating on the technical means of improving total information availability, largely ignoring organizational effects in any particular context. The high degree of correspondence shown between KM functional requirements and RIPS capabilities indicates that the RIPS will provide a powerful test bed for validating KM's organizational and management concepts.

---

20. M. Stonebraker: "Implementation of Integrity Constraints and Views by Query Modification," *Proc. ACM SIGMOD International Conference on Management of Data*, San Jose, California, May 1976, pp 65-78, (ed. W. F. King).

13. M. Stonebraker, E. Wong, and P. Kreps: "The Design and Implementation of INGRES," *ACM Transactions on Database Systems*, Vol 1, No. 3, September 1976, pp 189-222.

Integrity assertions and authorization constraints have received con-
siderable attention by other workers, which appears sound. However, we

REFERENCES

1. James F. Berry and Craig M. Cook: *Managing Knowledge as a Corporate Resource*. Contract Source Document, Version 4.5, 28 May 1976.

2. James F. Berry and Craig M. Cook: *Viewing Knowledge as a Resource in Federal Departments of the U.S. Government*. Economic Research Service, U.S. Department of Agriculture, September 1977.

3. M. E. Senko et al.: "Data Structures and Accessing in Database Systems," *IBM Systems Journal*, No. 1, 1973, pp 30-93.

4. L. S. Schneider and C. R. Spath: "Quantitative Data Description," *Proc. ACM SIGMOD International Conference on Management of Data*, San Jose, California, May 1975, pp 167-195 (ed. W. F. King).

5. M. E. Senko et al.: *A Data-Independent Architecture Model 1: Four Levels of Description from Logical Structures to Physical Search Structures*. IBM Research Report RF 982, February 1972.

6. L. S. Schneider and T. W. Connolly: "Generalized Data Base Management System Simulator," *Proc. 1976 Winter Simulation Conference*, Vol 2, December 1976 (ed. H. J. Highland, et al.).

7. Martin Marietta Database Research Project: *GDMS Math Model Simulator, Functional Specification, Design Specification and User's Guide*. NASA Contract Documentation, NAS9-13951, Johnson Space Center, Houston, Texas, September 1975.

8. Martin Marietta Database Research Project: *GDMS Real-Time Simulator, Functional Specification, Design Specification, and User's Guide*. NASA Contract Documentation, NAS9-13951, Johnson Space Center, Houston, Texas, September 1975.

9. Martin Marietta Database Research Project: *Data Dictionary Research*. NASA Contract Documentation, NAS9-13951, Johnson Space Center, Houston, Texas, September 1975.

10. E. F. Codd: "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, Vol 13, No. 6, June 1970, pp 377-387.

11. L. S. Schneider: "A Relational View of the Data Independent Accessing Model," *ACM SIGMOD International Conference on Management of Data*, Washington, D.C., June 1976, pp 75-90 (ed. James B. Rothnie).

12. Morton M. Astrahan and Donald D. Chamberlain: *Implementation of a Structured English Query Language*. RJ1464, IBM Research Center, San Jose, California, October 28, 1974.

13. M. Stonebraker, E. Wong, and P. Kreps: "The Design and Implementation of INGRES," *ACM Transactions on Database Systems*, Vol 1, No. 3, September 1976, pp 189-222.

14. M. M. Zloof: "Query by Example," *Proc. National Computer Conference*, AFIPS Press, Vol 44, 1975, pp 431-438.

conceptual.

15. E. F. Codd: "A Database Sublanguage Founded on the Relational Calculus," *Proc. 1971 ACM SIGFIDET Workshop on Data Description, Access, and Control*, San Diego, California, November 1971, pp 35-68.

16. E. Allman, M. Stonebraker, and G. Held: "Embedding a Relational Data Sublanguage in a General-Purpose Programming Language," *ACM SIGPLAN Notices*, Vol II Special Issue, Salt Lake City, Utah, March 1976, pp 25-35.

17. ANSI/X3/SPARC Study Group: *Database Management Systems, Interim Report.* FDT 7, No. 2, ACM, New York, 1975.

18. C. R. Spath and L. S. Schneider: "A Generalized End-User Facility for Relational Database Systems," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

19. L. S. Schneider: "A Relational Query Compiler for Distributed Heterogeneous Databases," Submitted for publication in *ACM Transactions on Database Systems*, January 1977.

20. M. Stonebraker: "Implementation of Integrity Constraints and Views by Query Modification," *Proc. ACM SIGMOD International Conference on Management of Data*, San Jose, California, May 1976, pp 65-78 (ed. W. F. King).

21. Masaharu Mizumoto, Motohide Umano, and Kokichi Tanaka: "Implementation of a Fuzzy-Set-Theoretic Data Structure System," Presented at Third International Conference on Very Large Databases, Tokyo, Japan, October 1977.

22. Dana Scott: *The Lattice of Flow Diagrams.* Technical Memo No. PRG-3, Programming Research Group, Oxford University Computing Laboratory, 45 Banbury Rd, Oxford, England.

23. Dana Scott: "Logic and Programming Languages," *Communications of the ACM*, Vol 20, No. 9, pp 634-641, September 1977.

24. D. S. Scott: "Data Types as Lattices," *SIAM Journal on Computing*, 5, 1976.

25. S. S. Yau and H. S. Fung: "Associative Processor Architecture--A Survey," *ACM Computing Surveys*, Vol 9, No. 1, March 1977, pp 3-28.

26. G. A. Anderson and R. Y. Kain: "A Content-Addressed Memory Design for Data Base Applications," *Proc. 1976 International Conference on Parallel Processing*, IEEE, 1976, pp 191-195.

27. D. L. Slotnick: "Logic per Track Devices," *Advances in Computers*, Vol 10, Academic Press, New York, 1970, pp 291-296.

28. C. Y. Lee and M. C. Paull: A Content Addressable Distributed Logic Memory with Applications to Information Retrieval," *Proc IEEE*, 15, June 1963, pp 924-932.

29. David K. Hsiao and Stuart E. Madnick: "Database Machine Architecture in the Context of Information Technology Evolution," *Proc. Third International Conference on Very Large Databases*, Tokyo, Japan, October 1977.

GLOSSARY OF TERMS

AQR	-	associated qualification rate
CAI	-	computer-aided instruction
CAM	-	content-addressable memory
CPU	-	computer processing unit
DBA	-	database administrator
DBMS	-	database management system
DBTG	-	database task group
DD/D	-	data dictionary/directory
DDL	-	data description language
DIAM	-	data-independent accessing model
DML	-	data manipulation language
DMS	-	data management system
EA	-	enterprise administrator
EUF	-	end-user facility
FKS	-	factual knowledge subsystem
FSTDS	-	fuzzy-set-theoretic data structure
GDBMS	-	generalized database management system
GDMS	-	generalized database management system
GEUF	-	generalized end-user facility
ID	-	identification
IS	-	information structure
JSS	-	judgement support subsystem
KBPA	-	knowledge-based personal assistant
KM	-	knowledge management
KRC	-	knowledge resource center
MIS	-	management information system
MMS	-	math model simulator
PACER	-	Program-Assisted Console Evaluation and Review
PKS	-	procedural knowledge subsystem
QC/T	-	query compiler/translator
QDD	-	quantitative data description
R	-	requirement

RDAL	-	representation-dependent accessing language
RDL	-	representation-dependent language
RIAL	-	representation-independent acessing language
RIPL	-	representation-independent programming language
RIPS	-	representation-independent programming system
ROM	-	rough order of magnitude
RTS	-	real-time simulator
TCS	-	translation and control subsystem
TNF	-	third normal form
TSO	-	time-sharing option

MISSION
of
Rome Air Development Center

RADC plans and conducts research, exploratory and advanced development programs in command, control, and communications (C^3) activities, and in the C^3 areas of information sciences and intelligence. The principal technical mission areas are communications, electromagnetic guidance and control, surveillance of ground and aerospace objects, intelligence data collection and handling, information system technology, ionospheric propagation, solid state sciences, microwave physics and electronic reliability, maintainability and compatibility.

F

3