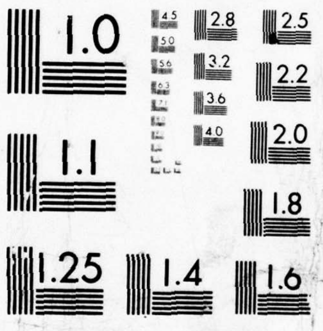
AD-A069 470 MARTIN MARIETTA AEROSPACE DENVER COLO DENVER DIV F/G 9/2
MODELING A LARGE SCALE DATA BASE.(U)
APR 79 T W CONNOLLY F30602-77-C-0142
UNCLASSIFIED MCR-78-1405 RADC-TR-79-47 NL

1 OF 1

AD A069470

END
DATE
FILMED
7-79
DDC


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

RADC-TR-79-47
Final Technical Report
April 1979

MODELING A LARGE SCALE DATA BASE

Martin Marietta Aerospace

T. W. Connolly

ROME AIR DEVELOPMENT CENTER
Air Force Systems Command
Griffiss Air Force Base, New York 13441

RADC-TR-79-47 has been reviewed and is approved for publication.

APPROVED: [illegible]

PATRICIA M. LANGENDORF
Project Engineer

APPROVED: [illegible]

HOWARD DAVIS
Technical Director
Intelligence and Reconnaissance Division

FOR THE COMMANDER: [illegible]

JOHN P. HUSS
Acting Chief, Plans Office

62702F

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

(19) REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER (18) RADC-TR-79-47 / 2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) MODELING A LARGE SCALE DATA BASE.	5. TYPE OF REPORT & PERIOD COVERED (9) Final Technical Report
	6. PERFORMING ORG. REPORT NUMBER (14) MCR-78-1405
7. AUTHOR(s) (10) T. W. Connolly	8. CONTRACT OR GRANT NUMBER(s) (15) F30602-77-C-0142
9. PERFORMING ORGANIZATION NAME AND ADDRESS Martin Marietta Aerospace P. O. Box 179 Denver CO 80201 Contract to	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (16) J. O. 45941026 (17) 10
11. CONTROLLING OFFICE NAME AND ADDRESS Rome Air Development Center (IRDA) Griffiss AFB NY 13441	12. REPORT DATE (11) April 1979
	13. NUMBER OF PAGES 60
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) Same (12) 65p.	15. SECURITY CLASS. (of this report) UNCLASSIFIED
	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: Patricia M. Langendorf (IRDA)

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

data bases, mathematical models, simulation

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

This document reports the methods and results of a study of a large database through use of modeling concepts and simulation software. The simulation software had been developed using the concepts of the four-level structure of the Data-Independent Accessing Model (DIAM) and the nomenclature and concepts of the Relational Model.

The study was undertaken to assess the feasibility of applying these concepts to modeling a large existing database--in this case, the USAF's

DD FORM 1 JAN 73 1473 UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

403225 ~~403225~~ 79 06 05 053

20) PACER database--and to demonstrate the use of existing components of query compiler software used to implement the concepts.

The report describes a model of the information content of a subset of the PACER database; the model is expressed as n-ary relations. In a subsequent phase of the study, the investigators developed a model of the access paths in the PACER system. The model uses the "strings" of the DIAM descriptive technique.

The information and access path models were coded as input to the simulation software, and the simulation was executed. For each of 46 input queries, the execution compiled a tabulation of all possible paths to the needed data so that the relative efficiency of the paths could be compared.

Based on analysis of the results, the author concludes that DIAM concepts enhanced by Relational Model nomenclature can serve well as the basis for study of large databases and that prototype software correctly compiles all paths applicable to an input query.

SUMMARY

This study was undertaken to assess the feasibility of applying the Relational Model in conjunction with Data-Independent Accessing Model I (DIAM I)[1] to a large existing database--the U.S. Air Force's Program-Assisted Console Evaluation and Review (PACER) database--and to demonstrate the use of existing components of query compiler software, part of the Representation-Independent Programming System (RIPS), developed by Martin Marietta.

The first phase of the study was familiarization with and analysis of the PACER database using government-furnished documents supplemented by conferences with a pertinent government contractor. The security classification of information required special attention because our work and reports were to be unclassified; the matter was successfully resolved.

In a second phase, we developed a model of the information content of a subset of the PACER system using n-ary relations. Such a model represents a canonic view of this information, independent of its storages, and forms the basis upon which general queries can be formulated without knowledge of the storage techniques.

In a third phase, we developed a model of the access paths actually implemented in the PACER system. This model is in terms of the "strings" of the DIAM I description as modified by Schneider.[2] It models the implemented paths among various data types as represented by relations.

Descriptions of the relational and access path models were coded as input to existing RIPS components, and software was executed in a simulation mode (i.e., no physical links existed between the contractor's software and the PACER database or users--these were represented by simulation). A set of queries was declared to exercise the simulated system. For each query, software analyzed the model and compiled a tabulation of all possible paths to the needed data. The paths were translated into an equivalent form at a procedural level directed toward individual data sets.

Based on analysis of the results, we concluded that the DIAM I methodology as modified by Schneider[2] can serve well as the basis for interfacing to large and complex databases and that the prototype software can correctly compile paths and equivalent translations at the access path level.

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

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

EVALUATION

This report demonstrated the feasibility of developing a relational model of the Program-Assisted Console and Review System at SAC. This will enable the power of the relational calculus to be applied to PACER data exploitation, and will be used in the design of the PACER follow-on system currently referred to as system 80. The technology proven under this contract will also be applied to effective accessing of disparate data bases.

[illegible]

PATRICIA M. LANGENDORF
Project Engineer

CONTENTS

INTRODUCTION 5
Purpose 5
Scope 5
PLAN FOR STUDY AND EXPERIMENTATION 5
Objectives 5
Tasks 5
ANALYSIS OF PACER SYSTEM (TASK 4.1.1) 7
Overview 7
Analysis of System 7
Security Considerations 10
RELATIONAL MODEL (TASK 4.1.2) 11
Concepts and Guidelines 11
General Approach 12
Detailed Approach 13
Resulting Relational Model 16
Implications 22
ACCESS PATH MODEL (TASK 4.1.3) 23
Concepts and Guidelines 23
General Approach 24
Detailed Approach 24
Resulting Model 33
Implications 33
DEMONSTRATION VIA QUERY COMPILER SOFTWARE (TASK 4.1.4) 38
Introduction 38
Specification of the Model 39
Specification of Traffic Load 39
Results of Execution 40
Evaluation of Results 40
CONCLUSIONS 54
Applicability of Relational Model 54
Applicability of Access Path Model 54
Applicability of Existing Components of RIPS Software 55
Requirements on Information Gathering 56
REFERENCES 57
APPENDIX A - A NOTE ON EVOLVING TERMINOLGY 58
GLOSSARY OF TERMS 60

FIGURES

1. Early version of PACER database structure 8
2. Recent version of PACER database structure 9
3. Relations modeled as subset of PACER database 17
4. Typical query for simulated database 22
5. P-string over attributes 24
6. Multiple PSGs over a relation 25
7. Use of the R-string 26
8. Modeling of master/detail chain 27
9. Modeling of "linked-to-master" chain 27
10. Model of working directory 28
11. Orgainization of PACER directories 29
12. Argument/master directory organization 30
13. Simple modeling of argument/master directory 31
14. More elaborate modeling of argument/master directory 31
15. Model of CALC entry 32
16. String path diagram of modeled PACER access paths 34

TABLES

1. DIAM PACER demonstration relational model 18
2. Detailed specification of relation (sample from Report 02) . . . 21
3. DIAM PACER demonstration string definitions and parameters . . . 36
4. DIAM PACER demonstration query tabulation 41
5. DIAM PACER path expansion (sample) 44
6. DIAM PACER demonstration RDAL for a single query (Report 12) . . 49
A-1. Equivalent terminology among various models 59

INTRODUCTION

Purpose

The purpose of this report is to report the work and results of Tasks 4.1.1 through 4.1.4 of Contract F30602-77-C-0142.

Scope

In this report, we describe our approach to understanding and analyzing the Program-Assisted Console Evaluation and Review (PACER) database, modeling of the database in terms of relations and string paths, execution of these models on our Query Compiler software, and our conclusions.

PLAN FOR STUDY AND EXPERIMENTATION

Objectives

The technical objectives of this study were:

1) Demonstrate the application of the Relational Model to the information content of a large complicated database. Show the process and techniques by which the model can be derived from available descriptive information.

2) Demonstrate the application of the string model of DIAM I with subsequent extensions[2] to modeling access paths of a large complicated database. Again, show the process and techniques by which the model can be derived.

3) Demonstrate existing Program-Independent Programming System (RIPS) software components in the environment of a large complicated database.

4) Assess the feasibility of RIPS as the baseline for a large-scale prototype system for interfacing with one or more existing systems concurrently, without users requiring knowledge of how data are distributed or stored.

Tasks

The following technical tasks were defined in the contractual statement of work:

1) Task 4.1.1 - Analyze the PACER data semantics and Data Access Path Network. Using unclassified documentation provided by the government, the contractor shall become sufficiently familiar with PACER data

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

elements, relationships, and cardinalities and access-path-level implementation to perform the subsequent activities. Reasonable hypothetical substitutions will be permitted as required to overcome either security restrictions or deficient documentation. A reasonable amount of consultation during this task will be provided by the government.

2) Task 4.1.2 - Develop a Quantified Relational Model. The collective PACER databases will be defined as a third-normal-form system of N-ary relations including underlying domain definitions. The normalization will be semantic rather than algorithmic to avoid the need for access to actual PACER data occurences. The results will be prepared as computer-readable input in the format prescribed by existing prototype software. The model will then be extended with the minimally sufficient quantitative parameters but at least including relation cardinality, single-attribute independent mapping distributions, and domain populations. These parameters will be prepared analogously. Relation definitions and quantitative characteristics will be compiled by the prototype software, and diagnostic-free compilation outputs will be considered evidence of task completion. Software modifications will be made as required.

3) Task 4.1.3 - Develop PACER Access Path Models. PACER access path networks will be defined as systems of DIAM I strings, including all pertinent string parameter declarations. Results will be prepared as computer-readable input to the format prescribed by the prototype software, and the string definitions will be compiled. Diagnostic-free compiler output will be evidence that the task is complete. Software modifications will be made as required.

4) Task 4.1.4 - Execute Decomposition Software and analyze results. A reasonable set of test queries will be hypothesized and expressed in the query language recognized by the prototype software. These will be prepared as computer-readable input in the format prescribed by the prototype software and submitted in batch mode for execution against the previously compiled definitions. Successful run completion and a diagnostic-free string traversal enumeration report will be evidence that the task is complete. Software modifications will be performed as required. Enumerated traversals for each test query will be examined and an analysis showing whether each would have yielded a correct answer will be prepared. In addition, both individually and in the ensemble, an analysis of query decomposition costs versus expected query processing costs will be prepared. These results, as well as technical explanations and recommendations as required, will be assembled and submitted as the final report.

ANALYSIS OF PACER SYSTEM (TASK 4.1.1)

Overview

The contract technical monitor designated the Program-Assisted Console Evaluation and Review (PACER) system as the database system to be used as the object of modeling. PACER is a U.S. Air Force information system that is both large and complex. Figure 1 is a sketch of an early version. Figure 2 indicates the complexity of a recent version.

Figure 1 provided our preliminary information about the system. Initial information furnished under the contract consisted of References 3 and 4. We found this information incomplete in several aspects. For supplemental information, we were directed to PRC Information Sciences Co. of Bellevue, Nebraska. From conferences and additional material,[5,6] they supplied us with sufficient information to allow us to continue our analysis. We wish to acknowledge the cooperative support of PRC personnel.

Analysis of System

For modeling, our analysis of the system was directed toward two different aspects. The first considered the information content of the database--what entities were represented in the database? What associations among these entities were of interest? How could these be cast in the formality of the relational model of data? These topics are discussed under Relational Model.

The second aspect considered implemented methods of accessing types of data from other types within the database and from external entries. We learned[6] that PACER was established under a variant of the IDS Data Management System termed the Reentrant Data Management System (RDMS). The main access methods supported are retrieval by "chain-walking" and hashing. A somewhat separate Directory Services System (DSS) provides means to establish and maintain a variety of sorted tables that can provide rapid access to data records. These topics are discussed under Access Path Model.

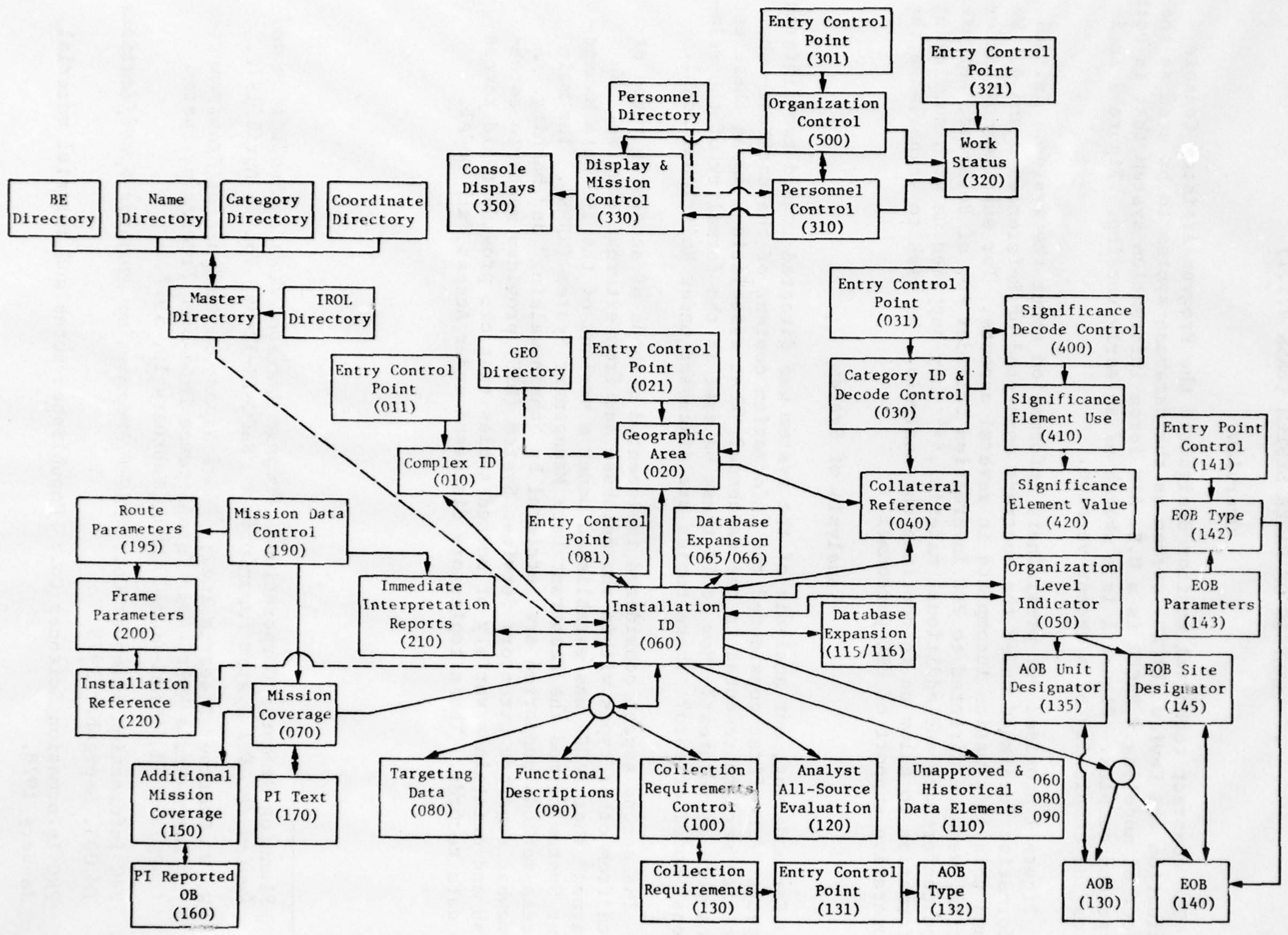
---

3. Planning Research Corporation: *Program Assisted Console Evaluation and Review (PACER) System Description.* RADC-TR-72-89, Vol 1, April 1972.

4. W. W. Gaertner Research Inc.: *Final Report on Analysis of Requirements for Large-Scale Multi-User Intelligence Data-Base Processing System (DBPS)* Vol II, WWGRI-4056-770210, February 10, 1977.

5. PRC Information Sciences Co.: *PACER Programming Language Specifications.* TM 003, September 1975.

6. PRC Information Sciences Co.: Unpublished notes and tutorial material, January 1978.



Entry Control Point (301)
Entry Control Point (321)
Personnel Directory
Organization Control (500)
Work Status (320)
Console Displays (350)
Display & Mission Control (330)
Personnel Control (310)
BE Directory
Name Directory
Category Directory
Coordinate Directory
Master Directory
IROL Directory
Entry Control Point (031)
Significance Decode Control (400)
Category ID & Decode Control (030)
Significance Element Use (410)
GEO Directory
Entry Control Point (021)
Entry Control Point (011)
Geographic Area (020)
Entry Point Control (141)
Complex ID (010)
Collateral Reference (040)
Significance Element Value (420)
EOB Type (142)
Route Parameters (195)
Mission Data Control (190)
Entry Control Point (081)
Database Expansion (065/066)
Immediate Interpretation Reports (210)
Organization Level Indicator (050)
EOB Parameters (143)
Frame Parameters (200)
Installation ID (060)
Database Expansion (115/116)
AOB Unit Designator (135)
EOB Site Designator (145)
Installation Reference (220)
Mission Coverage (070)
Additional Mission Coverage (150)
PI Text (170)
Targeting Data (080)
Functional Descriptions (090)
Collection Requirements Control (100)
Analyst All-Source Evaluation (120)
Unapproved & Historical Data Elements (110)
060
080
090
PI Reported OB (160)
Collection Requirements (130)
Entry Control Point (131)
AOB Type (132)
AOB (130)
EOB (140)

Figure 1. Early version of PACER database structure.

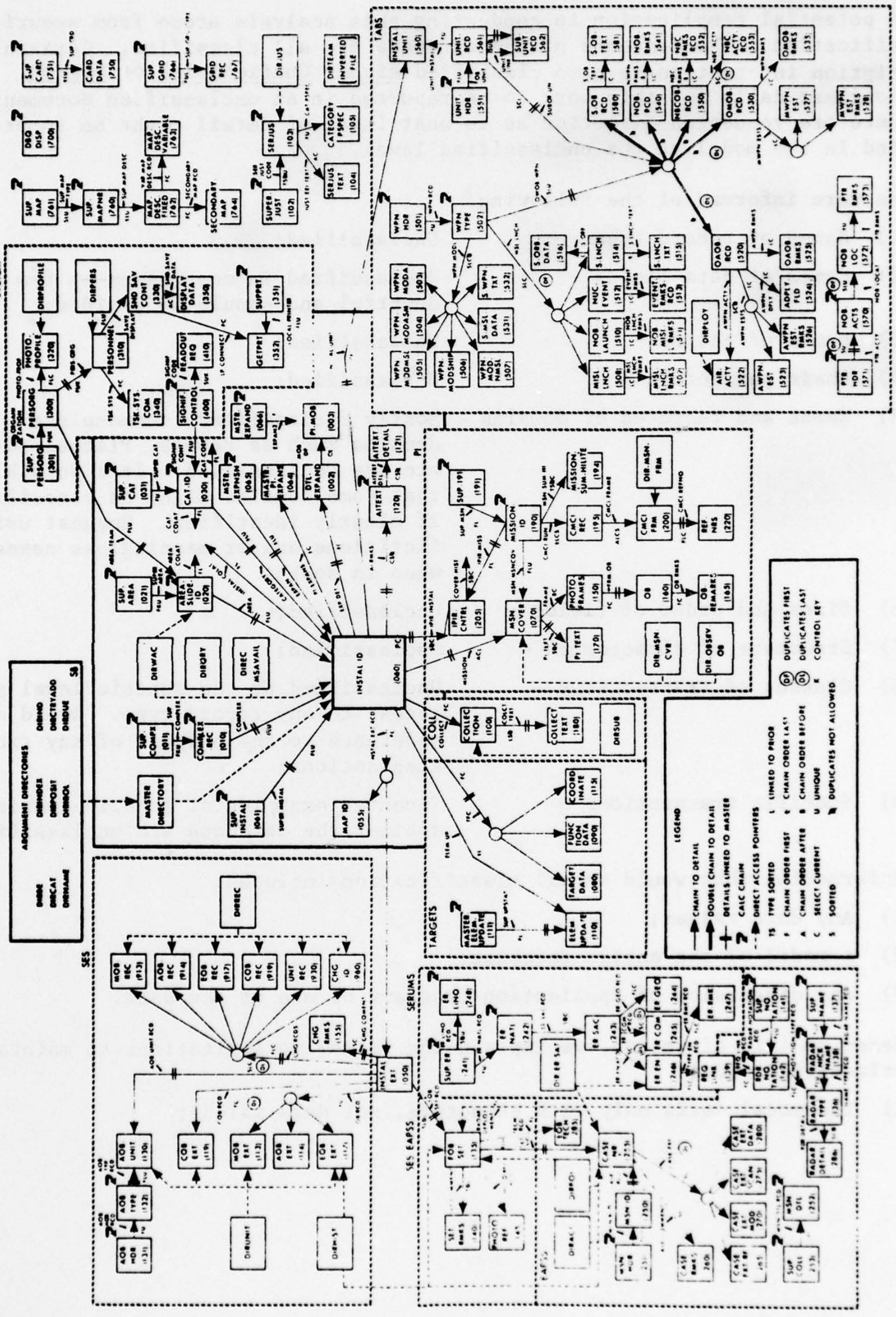


Figure 2. Recent version of PACER database structure.

Security Considerations

A potential complication in conducting this analysis arose from security classification. The contents of the database are all classified. Certain description information is also classified either Confidential or Secret. The contract calls for this work to be reported in an unclassified document. We therefore requested direction as to what level of detail might be incorporated in the model at the unclassified level.

We were informed of the following:

1) Names of record types -	Unclassified;
2) Names of data fields -	Unclassified in most cases--a few are doubtful and should be avoided;
3) Names of chains -	Unclassified;
4) Chain parameters -	Unclassified;
5) Names and features of domains -	Mostly Unclassified; commonplace domains such as People, Places, Dates, etc are Unclassified. It's possible that some domains might be classified if clearly identified. Suggest using fictitious and/or meaningless names when in doubt;
6) Sizes and order of fields -	Unclassified;
7) Structure of directories -	Unclassified;
8) Classes of transactions -	Unclassified at the generic level as access to any record type. Avoid any reference to the purpose of any true transactions;
9) Specific transactions -	Invented examples of generic queries against the database are unclassified.

Information that would entail classification includes:

1) Any data values;

2) A model of the entire database;

3) Any discussion of application programs or use of the data.

Based on this direction, we imposed the following limitations to maintain an unclassified model:

1) The model deals only with structure, not data values;

2) The model is limited to a subset of the total PACER system;

3) Attribute (field) names have been omitted, changed, and added in various cases;

4) Domain names for underlying populations have been taken as commonplace sets or as simple plurals of the attribute names. We avoid asking about the applications semantics. Such information would be an essential part of a complete analysis leading to a proper database interface;

5) To illustrate modeling of various access methods, transactions to exercise the interface model are taken as retrievals against the various relations declared.

RELATIONAL MODEL (TASK 4.1.2)

Concepts and Guidelines

Canonic Model - Quantitative comparison of various data management systems requires a common baseline. Similarly, interfaces among various distributed heterogeneous databases are most readily described and accomplished on such a common baseline.

For this study, we adopted the *relational data model* as the common canonic baseline for describing the information content of a database. The model provides a mathematically based foundation for information content and provides the basis upon which queries (transactions) can be stated.

Elements of Relational Model - The relational model was first published by E. F. Codd.[7] We summarize here the major elements of his approach; it is our intent to be desciptive rather than rigorous.

The Relation - Although the true relation is a formal mathematical structure, it is most often described as a table. The relation name corresponds to the table name. Each row of the tables is a tuple. Each column of the table is given a header name called an *attribute*. A tuple may not be duplicated; tuples in a relation have nc defined order. The order of columns is arbitrary. In the database, we attempted to establish a relation corresponding to each major type of real-world entity of interest.

The Attribute - The attribute is a name for some fact we wish to associate with the entity of interest.

The Domain - The domain is the set of values from which attribute values are drawn. Several attributes from the same or different relations may acquire their values from the same domain.

---

7. 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.

For example, if there is a domain COLOR, then a relation AUTOMOBILE might have an attribute called EXTERIOR-AUTO-COLOR and another called INTERIOR-AUTO-COLOR from the same domain. Still another relation called HOUSE might have attributes called HOUSE-COLOR and TRIM-COLOR from the same domain.

Keys - A key is a combination of one or more attributes whose values are never duplicated within another tuple of the same relation. The term *identifier* is also used for the same purpose. There may be multiple keys for the same relation. For example, the relation AUTOMOBILE may have as a key VEHICLE-ID-NO and may also have as a key the combination STATE-OF-REGISTRATION and LICENSE-NO.

General Approach

The approach used to define a relational model for an existing database differs from the approach used for design of a new system. In the latter case, the most effective analysis looks to the true entities and associations of the real-world situation. This approach is seldom practical in the case of the existing database. Instead, the analyst must look to the implemented records and chains for clues to the underlying entities and to the associations of interest to the user.

A detailed study of the user's semantics and the meaning of the data would ordinarily accompany analysis of the implementation. Such a study was not appropriate in our case because of the security classifications involved. The semantics are often explicit or implicit in the applications programs that use the database. In our case, it was not feasible to examine them.

In this study, we relied on unclassified record names, field names, and sketches of implemented chains. Our goal was to produce a reasonable but not necessarily completely accurate model.

It was necessary to make assumptions to complete the description of the model. For example, it was frequently necessary to guess which attributes constituted an identifier. In some cases, we declared a new attribute for this purpose. It was also necessary to assume domains from which the attributes came.

The definition of a relation in Codd's normal form demands that every nonidentifier attribute must depend on the identifier and on no other attribute(s). There is no way to confirm or reject this property for a group of arbitrarily named attributes. We therefore assumed that dependency conditions are met. In a practical case, any misunderstanding of such dependencies might result in anomalies when interacting with the database. Because the scope of this task dealt with retrievals, there was no potential for damage to the integrity of the database. However, even in an unsimulated case, there is still a potential for incorrect answers. A complete design to permit all operations including updates would demand a thorough understanding of the true dependencies to preserve database integrity.

Detailed Approach

Analysis of Records - Our basic method was to consider each record type and regard it as a candidate for direct redefinition as a relation. This direct translation is possible for a limited number of records, but most record types require special treatment as described in the following sections. Those found to need no special treatment can be translated directly.

Records with No Corresponding Relation - The PACER system contains a number of declared record types that are used not as information-bearing collections but as entry points for "chain walking." Examples of this use include record types SUP-AREA and SUP-COMPX. These have been describesd as one-of-a-kind records stored and accessed via hashing ("calc") of a known content word as a means to gain access to a chain of records of another type.

We do not include individual record types of this nature in the relational model, but consider them under the study of access paths.

Records Consolidated to a Single Relation - We believe that the PACER system contains portions in which instances of one record type are in one-to-one correspondence with those of another type. Both refer to the same real-world object, but the data have been separated into different record types for convenience of data processing.

In the relational model, record types are consolidated in a single relation that contains attributes corresponding to all data elements in the original record types. For example, our analysis led us to conclude that records dealing with maps were actually (or at least could be) in one-to-one correspondence. The record types were:

1) SUP-MAPNR;

2) MAP-DESC-FIXED;

3) MAP-DESC-VARIABLE;

4) SECONDARY-MAP.

We therefore consolidated these in a single relation to which we arbitrarily gave the name SUP MAPNR.

It is important to recognize that consolidation in the same relation at the information level *does not imply* that access paths or physical storage have been changed. The latter are considered at lower levels of the modeling process; no options have been precluded by the relational consolidation.

Records in Many-to-One Association - Most master-detail chains of the system suggest a many-to-one relationship.* For example, each organization has many personnel assigned.

---

* Some other database management systems pe⁻mit the use of "repeating groups" within a record definition. These are also many-to-one, and in such a case, it is necessary to declare a new relation corresponding to the contents of the "repeating group" and include a conjugate key with the parent

Such associations are readily modeled in the relational view. However, it is necessary that the conjugate key (e.g., Organization-Number) be included explicitly in the relation on the "many" side of the association. When the chain specification includes a particular MATCH-KEY, the association is quite clear, although implicit. The PACER programming language specification[5] states, "The use of this clause effectively defines the fields named as though they were contained in the detail record, although they are actually contained in the masters of the chain hierarchy." In the relational model, it is necessary to add the conjugate key to the "many" relation if it is not already declared.

The intent is more obscure when there is no specification of a MATCH-KEY but the association is to be done with "CURRENT" members of the master and detail record types. In such cases, linking criteria are vested in an applications program or an operator; in effect, the latter becomes part of the data management system. To satisfy the need of the relational model for an explicit conjugate key, we have selected or appended such an attribute by conjecturing what the applications program might be doing. Here again, we have tried to do our conjecturing at an unclassified level.

Records in Many-to-Many Association - By examining the implemented chains, we deduced that some associations are of the many-to-many type. For example, the INSTAL UNIT chain suggests that a single installation may have several units, and that the same unit may occur at multiple installations. In the relational "flat file" concept, the solution to this case is to establish a new relation that serves to show which instances of installation are associated with which instances of unit. The population of the new intermediate relation is potentially as large as the product of the populations of the two original relations but is unlikely to be that large in practice.

Example - To illustrate and apply the use of many-to-many and many-to-one associations, we examined associations appropriate to the PERSONNEL relation and observed the modifications necessary. For illustrative purposes, we have taken a few liberties with the interconnections. We assumed that the desired associations are as sketched at the top of the next page.

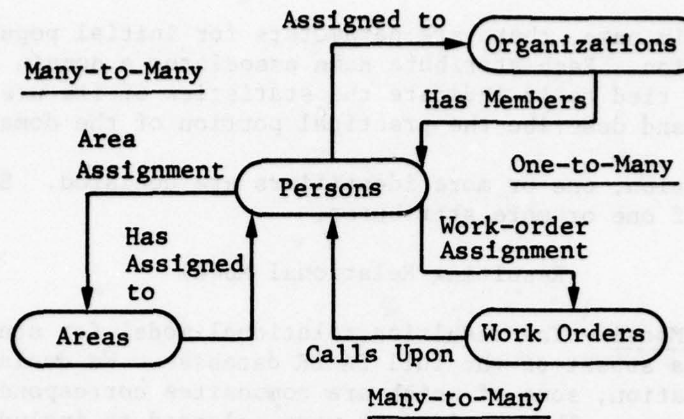
The sketch illustrates the assumptions that:

1) A person may be assigned to only one organization; an organization may have many persons assigned;

2) A person may be assigned to work on more than one area team; an area team may consist of many persons;

3) A person may be assigned to work on more than one work order; a work-order team may consist of many persons.

5. PRC Information Sciences Co.: *PACER Programming Language Specifications*. TM 003, September 1975.



A resulting relational model to account for these is then:

```
PERSONNEL (NAME,..........      ,ORGNO)   Conjugate key included
ORGANIZATIONS (ORGNO,...,.....)           explicitly.
AREAS (AREANR,...,.....)
WORKORDERS (WONR,...,....,)
AREATEAM (AREANR, NAME )
                        }  <-------  New intermediate relations
WOTEAM (WONR, NAME)                   defined.
```

Again we emphasize that this modeling at the information level does not preclude an eventual implementation by chain ponters or otherwise.

Attributes and Domains - Relation names and names of their included attributes were established as described above. The relational model also requires the naming of domains from which the attributes are taken. In effect, an attribute name establishes an association between a domain name and a relation name and tells what role the elements of the domain play. (Recall that in the equivalent Entity Set Model, the "Role Name" is the tie between the "Entity Set Name" and the "Description Set Name," as shown in Appendix A.

It was necessary to name and specify about 200 domains with which the various attributes were associated. These include such categories as DATES, PEOPLEA, and PEOPLEB. The latter two illustrate the point that the domain is to contain the *names* or other representations of the object, not the object itself, and that there may be different dissimilar sets of names. The basic criterion for assigning two different attributes to the same domain is that a direct comparison is meaningful in that case.

Other Parameters - Complete specification of the relational model for simulation requires and/or permits the use of many parameters in connection with each major entry. For example, with each relation name is specified an initial population. Also, average anticipated rates for retrievals, additions, deletions, and changes can be used by the simulation software in a query generation mode to simulate statistical traffic patterns; this mode was not exercised in our study.

For each domain name, there are parameters for initial population and encoding information. Each attribute name associates a domain and a relation. Parameters tied to it indicate the statistics of its use in random query generation and describe the practical portion of the domain used.

For each relation, one or more identifiers are declared. Each identifier is made up of one or more attributes.

Resulting Relational Model

Scope of the Model - The resulting relational model for study and experimentation is a subset of the full PACER database. We declared 29 relations for simulation, some of which are composites corresponding to two or three record types. These relations were selected to include examples of modeling techniques both for relations and for access paths described in a later section.

A number of original attribute names have been altered and some omitted in the interests of simplicity and retaining the unclassified nature of the study. For demonstration, none of these changes have a significant effect on the model.

Presentation of Model - Figure 3 is an overview of the relational model. The view intentionally omits any suggestion of access or interconnection; this is the subject of the Access Path Model section.

The relations and their assigned attributes are listed in Table 1, which is an output Report Number 23 from the simulation.

Details of the relations, their attributes, domains, and parameters are in Report Number 02 in Appendix B. In this report and others, the notation is that of the Entity Set Model, which was the original basis of the simulation software* The reader should interpret Description Set Names (DSNs) as relations, Entity Set Names (ESNs) as domains, and Role Names (RNs) as attributes. Table 2 is a sample page from Report Number 02.

A Basis for Queries - It should be remembered that the relational model is intended to correspond to the user's view of his information. It serves as a foundation for writing queries (which in this study include only retrievals).

The query language used in the simulator is called Representation-Independent Accessing Language (RIAL). In content, it is generally similar to other languages proposed for relation systems. Figure 4 is an example of a query.

* Refer to Appendix A on notation.

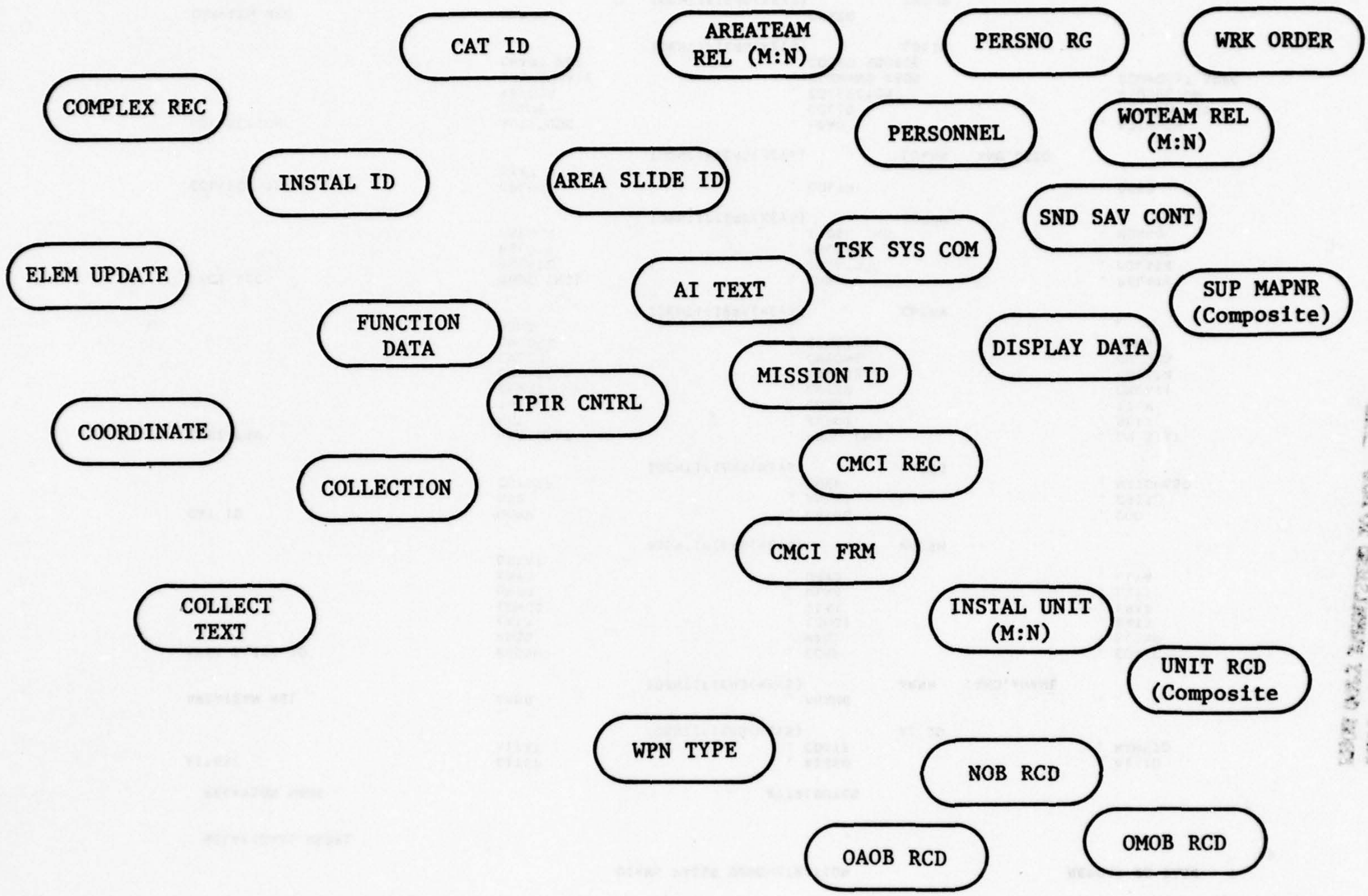


Figure 3. Relations modeled as subset of PACER database.

Table 1. DIAM PACER demonstration relational model.

```
                                      DIAMS PACER DEMONSTRATION                  REPORT 23 PAGE   1
  RELATIONAL MODEL

  RELATION NAME                                            ATTRIBUTES
 AITEXT                   AITYP                  , AISEQ                   , AI ID
                          AITXT                  , CDATE                   , NAMEID
                                IDENTIFIERS(KEYS)          AI ID

 AREATEAM REL             ARNR                   , ANAME                   ,
                                IDENTIFIERS(KEYS)          ARNR   .AND.ANAME

 AREA SLIDE ID            WACSH                  , EDNP                    , EDATE
                          PROD                   , MAG                     , SLDNP
                          LAT1                   , LONG1                   , LAT3
                          LONG3                  , LLX1                    , LRX2
                          URX3                   , ULX4                    , LLY1
                          LRY2                   , URY3                    , ULY4
                          CSTAT
                                IDENTIFIERS(KEYS)          WACSH

 CAT ID                   UGRP                   , CATEG                   , CGO
                          NTR                    , ANLST                   , CRIT
                          CUTOFF                 , COEF                    , MATCH030
                                IDENTIFIERS(KEYS)          CATEG

 CMCI FRM                 HOC STAT               , PROG IND                , CM STAT
                          HGT                    , ALPHA                   , BETA
                          THETA                  , CONE                    , FILM
                          SLANT                  , CMFRM                   , CMLAT1
                          CMLAT2                 , CMLA13                  , CMLAT4
                          CMLON1                 , CMLON2                  , CMLON3
                          CMLON4                 , CFMDATE                 , RUNN
                          MSNA                   ,
                                IDENTIFIERS(KEYS)          CMFRM

 CMCI REC                 PROG INDI              , CNRUN                   , PSLAT1
                          PSLAT2                 , PSLAT3                  , PSLAT4
                          PSLON1                 , PSLON2                  , PSLON3
                          PSLON4                 , T DEL IND               , MSNNO
                                IDENTIFIERS(KEYS)          CNRUN

 COLLECT TEXT             COLNR                  , COLTR                   , CSEQ
                          CTXT                   ,
                                IDENTIFIERS(KEYS)          COLNR  .AND.CSEQ

 COLLECTION               ALTITUDE               , LENG                    , AZIMUTH
                          INAME                  , COLID                   , ACTION REQ
                          READER                 , COLLECTOR               , REQUESTOR
                          SUBMSNDATE             , COMMAND REQS            , COMPOSIT SPEC
                          CHART REF              , COORD SOURCE            ,
                                IDENTIFIERS(KEYS)          COLID

 COMPLEX REC              CMAJR                  , CPXTR
                                IDENTIFIERS(KEYS)          CMAJR

 COORDINATE               GEMS CODE              , PRIORITY                , MTD DATE
                          CORDSTAT               , SCOORDS                 , UTMCOORDS
```

Table 1. (cont).

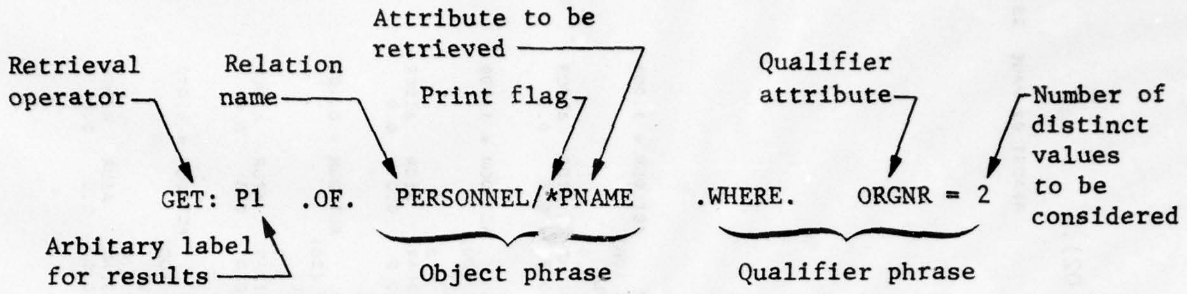
```
                                              DIAMS PACER DEMONSTRATION                  REPORT 23 PAGE   2
     RELATIONAL MODEL

                          METCOORDS                    , MTD DATA                     , NON ATMR COORDS
                          NSTDB COORDS                 ,
                                          IDENTIFIERS(KEYS)         UTMCOORDS

     DISPLAY DATA         SDID2                        , DISSEQ                       , DISIZ
                          DISDAT                       ,
                                          IDENTIFIERS(KEYS)         SDID2  .AND.DISSEQ

     ELEM UPDATE          EFLAG                        , DESIG                        , ELVAL
                          INS                          , PDATE                        , UP STAT
                          REFNUM                       ,
                                          IDENTIFIERS(KEYS)         REFNUM

     FUNCTION DATA        SIG                          , CAP                          , PCT
                          RMKS                         , INS                          , CNFD
                          CRNT                         ,
                                          IDENTIFIERS(KEYS)         SIG

     INSTAL ID            NAME                         , INTID                        , BE
                          BESUF                        , MAJ                          , MIN
                          CAT                          , COORD                        , VLC
                          WACSHEET                     , CTRY                         , IROL
                          ADDST                        , JN                           , ONC
                          SEQ                          , RPTDUE                       , STAT
                          CURSOURCE                    , INDIC                        , ELMT
                          TABSIND                      , NIC                          , OBJ
                          DIVAL                        , LOCIND                       , JADOB
                          NULIND                       , POSIT                        , ADMIN
                                          IDENTIFIERS(KEYS)         NAME
                                          IDENTIFIERS(KEYS)         INTID

     INSTAL UNIT          UI INS                       , UI UNIT                      ,
                                          IDENTIFIERS(KEYS)         UI INS .AND.UI UNIT

     IPIR CNTRL           BST FRM DATE                 , BST FRM TIME                 , FUN H
                          FRM H                        , ST H                         , URGX H
                          URGY H                       , WP H                         , INTRP H
                          INDEX H                      , H ACT                        , H STATS
                          ORCAM H                      , MSNN                         ,
                                          IDENTIFIERS(KEYS)         BST FRM DATE .AND.BST FRM TIME

     MISSION ID           PROJ                         , PROS IND                     , MSNNR
                          SDATE                        , FDATE                        , FOCAL
                                          IDENTIFIERS(KEYS)         MSNNR

     NOB RCD              NOID                         , NOTYP                        , NOOTHER
                          NOWTYP                       , NOUNIT                       ,
                                          IDENTIFIERS(KEYS)         NOID

     OAOB RCD             UTYP1                        , UTYP2                        , UTYP3
                          UUNIT                        ,
                                          IDENTIFIERS(KEYS)         UTYP2

     OMOB RCD             OMID                         , OMCAT                        , OMOTHER
                          OMTYP                        , OMUNIT                       ,
```

Table 1. (concl).

```
                                               DIAMS PACER DEMONSTRATION                REPORT 23 PAGE   3
  RELATIONAL MODEL
                                                  IDENTIFIERS(KEYS)          OMID
PERSNORG                          ORG                            , PRNTR
                                                  IDENTIFIERS(KEYS)          ORG
PERSONNEL                         PNAME                          , RANK                              , SSN
                                  PERSCLASS                      , TSKOV                             , PERSCODE
                                  ORGNR                          ,
                                                  IDENTIFIERS(KEYS)          PNAME
SND SAV CONT                      TYPDIS                         , LOCPRT                            , RCI
                                  SNDR                           , RCVR                              , SDDAT
                                  SDID                           , SDSUB                             , SDCON
                                  SDPRJ                          , SDSI                              ,
                                                  IDENTIFIERS(KEYS)          SDID
SUP MAPNR                         MAPID                          , PROJN                             , CONV
                                  SCALE                          , STD PAR                           , CTR
                                  LINE1                          , LINE2                             , LINE3
                                  MAG SLD                        , MAPBOT                            , MAPTOP
                                  MAPRT                          , MAPLFT                            ,
                                                  IDENTIFIERS(KEYS)          MAPID
TSK SYS COM                       REF                            , PNAME2                            ,
                                                  IDENTIFIERS(KEYS)          REF     .AND.PNAME2
UNIT RCD                          UID                            , UNAM                              , UMOD
                                  UOTHER                         , SUBUMOD                           , SUBUOTHER
                                                  IDENTIFIERS(KEYS)          UID
                                                  IDENTIFIERS(KEYS)          UNAM
WOTEAM REL                        WONRNR                         , WNAME
                                                  IDENTIFIERS(KEYS)          WONRNR .AND.WNAME
WPN TYPE                          WTYPA                          , WNAM                              , WOTHER
                                  WTYPB                          , WTYPC
                                                  IDENTIFIERS(KEYS)          WNAM
                                                  IDENTIFIERS(KEYS)          WTYPA
WRK ORDER                         WONR                           , RTIME                             , ASGN
                                  DUEDATE                        , WOTXT                             , PRTY
                                  WOSEQ                          , MATCH320                          ,
                                                  IDENTIFIERS(KEYS)          WONR
```

Table 2. Detailed specification of relation (sample from Report 02).

```
                                    DIAMS PACER DEMONSTRATION                                        REPORT 02 PAGE  25
DSN  =DISPLAY DATA
       IPS                PS          IRR            IAR            IDR            ICR
       0.50000E+03        0.0         0.86400E+05    0.60480E+06    0.60480E+06    0.0

   ESN  =DISPID                       IPS            LEN   DT   PS
                                      0.20000E+03      6   A    0.0

   ESN  =DISPSEQ                      IPS            LEN   DT   PS
                                      0.10000E+03      2   N    0.0

   ESN  =DISPSIZ                      IPS            LEN   DT   PS
                                      0.10000E+04      4   N    0.0

   ESN  =DISPDATA                     IPS            LEN   DT   PS
                                      0.40000E+03   2000   A    0.0

   RN   =SDID2                          ESN  =DISPID                           IPS =0.20000E+03    ACT DOM = 1.000
          MAPPING DISTR (MD)        ARRIVAL DISTR (AD)       RETRIEVAL DISTR (RD)       DEPARTURE DISTR (DD)
              TYPE = DEFAULT               TYPE = DEFAULT            TYPE = DEFAULT             TYPE = DEFAULT
         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
          ARR    AAR    ADR    ACR   AQRR   AQAR   AQDR   AQCR   AQRV   AQAV   AQDV   AQCV   AIAR  AICAR   AIDR  AIDCR
          0.500  1.000  1.000  0.0   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0   0.0    0.0   0.0
        NO DEPENDENT ENTITY SETS
   RN   =DISSEQ                         ESN  =DISPSEQ                          IPS =0.10000E+03    ACT DOM = 1.000
          MAPPING DISTR (MD)        ARRIVAL DISTR (AD)       RETRIEVAL DISTR (RD)       DEPARTURE DISTR (DD)
              TYPE = DEFAULT               TYPE = DEFAULT            TYPE = DEFAULT             TYPE = DEFAULT
         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
          ARR    AAR    ADR    ACR   AQRR   AQAR   AQDR   AQCR   AQRV   AQAV   AQDV   AQCV   AIAR  AICAR   AIDR  AIDCR
          0.500  1.000  1.000  0.0   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0   0.0    0.0   0.0
        NO DEPENDENT ENTITY SETS
   RN   =DISIZ                          ESN  =DISPSIZ                          IPS =0.10000E+03    ACT DOM = 0.100
          MAPPING DISTR (MD)        ARRIVAL DISTR (AD)       RETRIEVAL DISTR (RD)       DEPARTURE DISTR (DD)
              TYPE = DEFAULT               TYPE = DEFAULT            TYPE = DEFAULT             TYPE = DEFAULT
         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
          ARR    AAR    ADR    ACR   AQRR   AQAR   AQDR   AQCR   AQRV   AQAV   AQDV   AQCV   AIAR  AICAR   AIDR  AIDCR
          0.500  1.000  1.000  0.0   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0   0.0    0.0   0.0
        NO DEPENDENT ENTITY SETS
   RN   =DISDAT                         ESN  =DISPDATA                         IPS =0.40000E+03    ACT DOM = 1.000
          MAPPING DISTR (MD)        ARRIVAL DISTR (AD)       RETRIEVAL DISTR (RD)       DEPARTURE DISTR (DD)
              TYPE = DEFAULT               TYPE = DEFAULT            TYPE = DEFAULT             TYPE = DEFAULT
         0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0
          ARR    AAR    ADR    ACR   AQRR   AQAR   AQDR   AQCR   AQRV   AQAV   AQDV   AQCV   AIAR  AICAR   AIDR  AIDCR
          0.500  1.000  1.000  0.0   0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0    0.0   0.0    0.0   0.0
        NO DEPENDENT ENTITY SETS
   ID-S FOR THIS DESCRIPTION SET

        SDID2
        DISSEQ
```



Attribute to be retrieved
Retrieval operator
Relation name
Print flag
Qualifier attribute
Number of distinct values to be considered
GET: P1 .OF. PERSONNEL/*PNAME .WHERE. ORGNR = 2
Arbitary label for results
Object phrase
Qualifier phrase

Figure 4. Typical query for simulated database.

The meaning of the qualification phrase differs in the simulation case from the actual retrieval case. Whereas a retrieval in an actual system might state

```
.....     .WHERE.     COLOR = "RED" .OR. "BLUE"
```

the simulation query has

```
.....     .WHERE.     COLOR = 2
```

meaning to compute statistics based on two values.

Implications

What Can Be Modeled? - The study has demonstrated that it is feasible to model as relations collections of names of real-world entities and their attributes in a larger scale of complexity than demonstrated in the open literature. It is feasible to infer that implemented record types correspond in some way to relations and that their data fields will frequently bear a correspondence to definable attributes. However, there is no assurance that these correspondences are complete or do not contain dependencies that must be removed. Modeling an implemented system therefore requires a study of real-world entities, their associations, and the assumptions and processing used in applications programs.

We have demonstrated cases in which the implemented record types:

1) May be translated directly into a relation;
2) May be consolidated into a single relation;
3) May represent a hierarchical association that may require that implied attributes be made explicit;
4) May represent many-to-many associations that may require definition of an intermediate relation that does not corrspond to any record type;
5) May not be part of the relational model at all but instead represent an access path.

We have shown the techniques appropriate to modeling cases 1, 2, 3, and 4 at the relational level. Case 5 will be modeled at the access path level.

What is the Penalty for Incorrect Understanding? - The importance of the database relational model in connection with distributed heterogeneous databases is that it forms the canonic view via which external users of the database may make inquiries. If the system presents a user with a list of "relations" and "attributes" that contain dependencies or other deviations from fully normalized relations, such deviations could possibly result in loss of database integrity if they were used as the basis for update traffic. If interaction is limited to retrieval queries, the penalty is potentially incorrect answers.

ACCESS PATH MODEL (TASK 4.1.3)

Concepts and Guidelines

Basis of Model - As the basis of our access path model, we adopted the second level of DIAM--Senko's "string level" as modified by Schneider[8]--hereafter referred to as "Relational DIAM." Schneider defined three classes of "strings" that in combination could depict useful types of access paths among relations and corrsponding to operations of the relational algebra. These are described in detail in References 1 and 8. We summarize the three classes of strings:

1) Projection-string (P-string or PSG) - This string type connects some or all of the attributes within a single relation. It is depicted in diagrams as a circle.

2) Restriction-string (R-string or RSG) - This string type connects homogeneous sets of string types--either PSGs or other RSGs. Parameters of the string can specify subsetting, ordering, and/or an entry mechanism. Examples are shown under Restrictions over Relations.

3) Join-string (J-string or JSG) - Connects two dissimilar string types, such as two PSG types, a PSG and RSG, another JSG and a RSG, or other combinations. Examples are shown under Interrelation Access. A JSG is depicted as a triangle.

(In the computer model, figures, tables, and appendixes, strings have been given *names* that reflect the original terminology of DIAM as described in Appendix A. Strings are named as ASGs, ESGs, or LSGs corresponding to PSGs, RSGs, or JSGs, respectively.)

---

8. 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).

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

Lower Levels of DIAM Model - Below the string level in the relational DIAM model are the encoding level, frame level, and physical device level. We did not address these levels in this study. However, it is pertinent to recognize the content of these levels and their relation to the upper levels.

The encoding level describes how data are structured for storage and how associations are to be implemented. It is concerned with allocation of these to address spaces and the choice of techniques such as the use of pointers.

In modeling access paths based on an existing implementation, we must examine features that are at the encoding level to understand which paths are implemented in order to put them in the access path model. In the implementation, the distinction between levels is not always clear. We discuss this topic further under Relevance of Lower Levels of DIAM.

The frame level is concerned with where in the address spaces the data and paths are stored. The physical device level is concerned with the parameters of available storage devices and how the logical address space is mapped to the physical storage. The level also includes (as appropriate) the features of the computer's operating system that control actual access to devices.

General Approach

The general approach was to consider the access paths actually implemented and form models of them. Access paths of interest included paths to individual attributes, paths to relation instances, paths between relations, and paths for entry and access to various relation types.

Our intent was to attempt to model each implementation technique encountered and provide sample access paths. In the interests of simplicity, we avoided extensive duplication of techniques.

Detailed Approach

Simple Strings over Attributes - Attributes of a relation are connected by the projection string (PSG). In most relations, a single PSG connects every attribute of the relation. An instance of the string type corresponds to an instance of the relation.

The PSG can be depicted as shown in Figure 5. Attribute names are frequently omitted from the sketch.

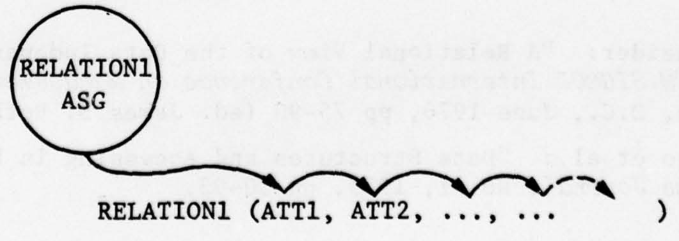


Figure 5. P-string over attributes.

Multiple Strings over Attributes - In some cases, attributes of a relation may be accessed by more than one PSG. This may occur when portions of the relation are actually distinct (e.g., corresponding to separate implemented record types) or when a particular attribute is accessed in different ways.

An example taken from the model involves the various attributes associated with a map. Recall that in the relational model, we consolidated several record types that were one-to-one. At the access-path level, we recognize that various groups of attributes from the relation are implemented together and are therefore accessed separately. This example is sketched in Figure 6.

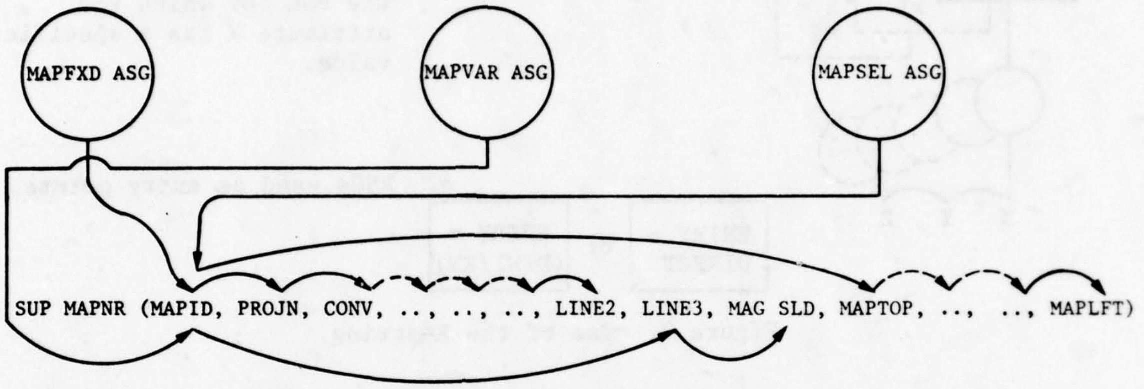

MAPFXD ASG
MAPVAR ASG
MAPSEL ASG
SUP MAPNR (MAPID, PROJN, CONV, .., .., .., LINE2, LINE3, MAG SLD, MAPTOP, .., .., MAPLFT)

Figure 6. Multiple PSGs over a relation.

Restrictions over Relations - The restriction string (RSG) represents a path to a homogeneous collection of instances of the PSG for a relation. It may serve any of the following purposes or a combination of them:

1) Connect all instances of a PSG for access; in this case, there is one instance of the RSG;

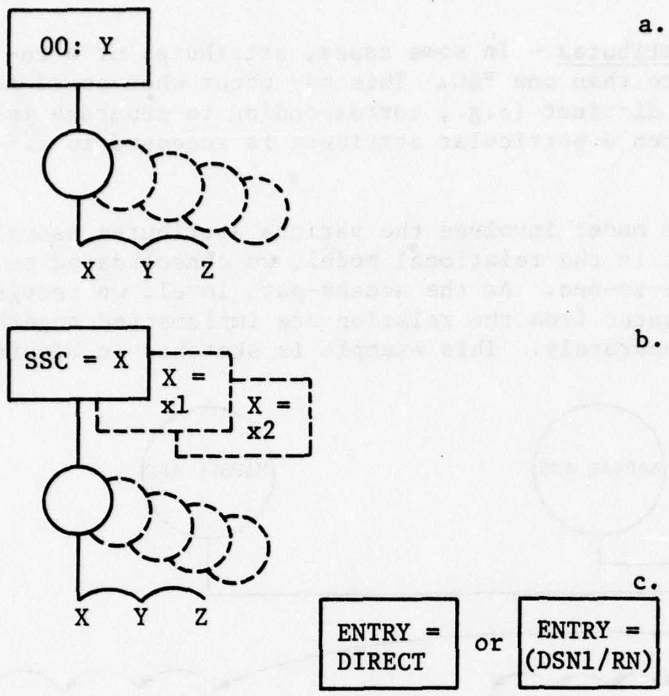
2) Perform subsetting (restriction) of the relation by connecting those with specified values of a particular attribute (or combination); in this case, there is an instance of the RSG for each value of the attribute;

3) Perform ordering of the instances under it, based on one or more attributes;

4) Specify an entry point at which a search may begin.

Examples sketched in Figure 7 illustrate typical uses of the RSG. The parameter SSC means "subset selection criterion," which specifies the basis for assigning instances of a PSG to an instance of the RSG; the parameter OO means "order on," which specifies the basis for ordering instances of the PSG under each instance of the RSG.

Interrelation Access - The join string (JSG) serves to specify a path between two relation types. It may connect differing string types--a PSG type and another PSG type, a PSG type and a RSG type, another JSG type and a RSG type, and so on.



a. A single instance of the RSG connects all instances of the RSG in a sequence ordered on the attribute Y.

b. Separate instances of the RSG connect instances of the PSG for which the attribute X has a specific value.

c. RSGs used as entry points.

Figure 7. Use of the R-string.

In general, there will be an instance of the JSG for each instance of the first string type on its exit list. In simple cases, there will be the same number of instances of the second string type on the exit list, but in practical cases the numbers may disagree.

A parameter of principal importance is the Matching Criterion (MC), which specifies how an instance of the first string type is to be matched to an instance of the second. The specification is stated by equating more attribute names from the first relation the JSG is "over" to one or more attributes of the second relation it is "over." For simulation, the parameter can also contain a numerical value expressing the condition that the population of the second string type differs from the population of the first.

Modeling of Chains - In the RDMS used for PACER, the means of interrecord access is the chain. Implementation of a chain is a series of pointers imbedded in the records. Declaration of a chain names a "master" record type and a "detail" record type. Each record of the master type contains a pointer to the first corresponding detail record. The detail record then contains a pointer to the next detail record on the chain. The final detail record contains a pointer back to the original instance of the master record. As shown in Figure 8, a simple chain is modeled using a JSG.

As an option, a chain may be "linked-to-master." In this case, *every* detail record contains a pointer to its corresponding master record, as modeled in Figure 9.

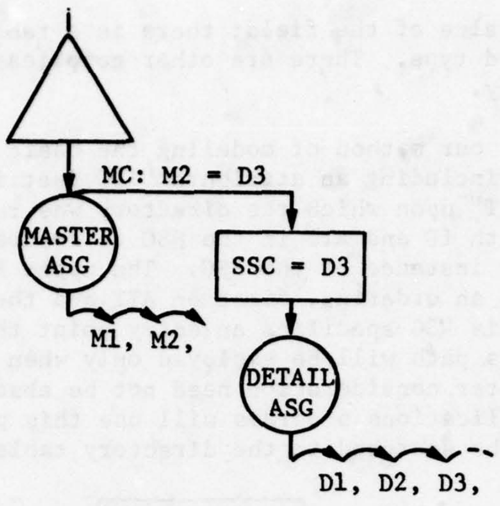


Figure 8. Modeling of master/detail chain.

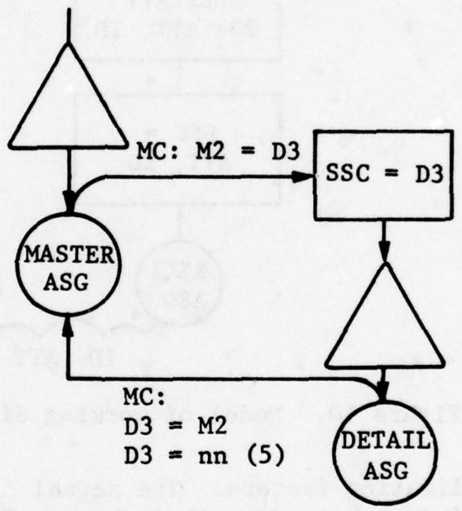


Figure 9. Modeling of "linked-to-master" chain.

<u>Modeling of Directories</u> - The PACER database uses a number of directories to access certain information more rapidly. For our purposes, these may be categorized as:

1) Working Directories;

2) Argument Directory/ Master Directory System.

Working Directories - First, we consider the working directories. In the PACER system, these are maintained by the user and applied in a number of imaginative ways. We will consider only the principal application of such a directory--to provide rapid access to a particular relation based on a particular attribute. The directory functions as an index over a primary or secondary key. Its basic implementation consists of a table in which each entry contains the value of a particular field from the record and an associated address (or equivalent) of the corresponding record instance. Entries are sorted on the value of the field; there is a table entry for each instance of the record type. There are other complicating features that we will discuss shortly.

Figure 10 shows our method of modeling the basic working directory. We assumed a relation including an attribute "ID" that is an identifier and another attribute "ATT" upon which the directory was to be built. The RSG over the PSG includes both ID and ATT in the SSC to assure that an instance of the RSG exists for each instance of the PSG. The upper RSG, which has a single instance, specifies an ordering, first on ATT and then on ID (within repeated values of ATT). This RSG specifies an entry point that is a function of ATT; in other words, this path will be employed only when the query is qualified upon ATT. (The latter consideration need not be absolutely true, but it is reasonable that applications programs will use this pathway only when entering with values to be searched in the directory table.)

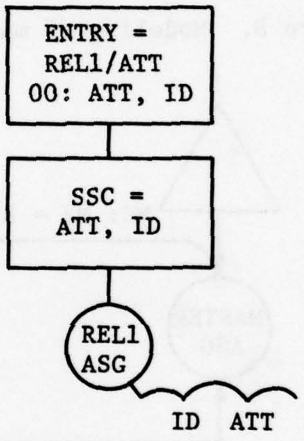


Figure 10. Model of working directory.

Now for the complicating factors. The actual directory mechanism goes beyond the basic implementation described above. Figure 11 is a sketch of the actual implementation.

Directory entries are stored on directory pages. As an aid to rapid search of the table, limit pages are maintained. Each entry on a limit page corresponds to the highest value of the sorted entries found on a specific directory page. The indexing is carried even a step further by a DOR2 table that in turn contains entries corresponding to the highest value on each limit page. In the Relational DIAM model, these types of information that deal with the allocation and use of address space (e.g., pages are modeled at the encoding level. Because our study did not include the encoding level, we have only an approximate model at the access path level. The number of accesses predicted will be in error, but in a known fashion.

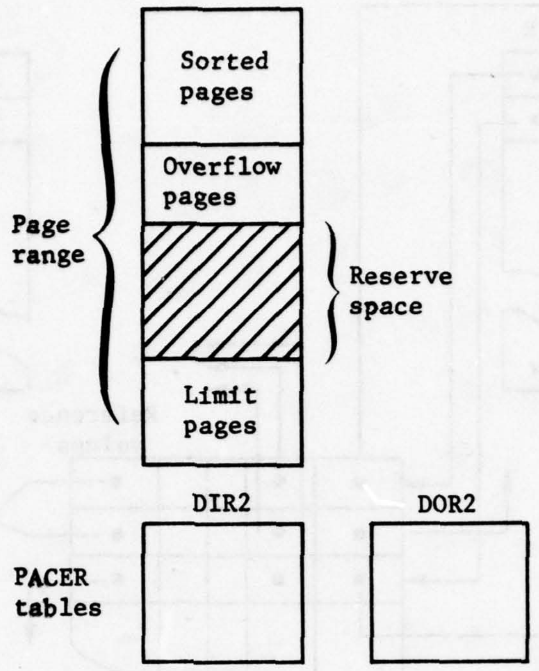


Figure 11. Organization of PACER directories.[6]

Another complication is the fact that the actual directory is made up of sorted pages and overflow pages. The latter contain, in unsorted form, entries that are additions or changes since the most recent batch consolidation and sorting. The important point is that a search must examine *both* portions of the directory; different search methods apply to each. The implication in the model is that some instances of a relation are accessed one way and other instances are accessed another. This is an example of "restriction-distribution," a term that applies to cases in which some elements of a relation may be stored in a file or at a site different from other elements. The concepts of restriction-distribution have been described by Schneider.[2] The corresponding additions do not exist in our present simulation software but are planned for inclusion in the future. It was not possible during this study to model the dual nature of the directories. Our demonstration model contains a representation of the "DIRPERS" directory as a sample of our method.

Argument/Master Directories - We now consider the Argument Directory/Master Directory System (Fig. 12). It involves a group of nine argument directories, each over one attribute and a common master directory. Each argument directory resembles the working directory previously discussed, except that each entry points not to the record itself but to an entry in the master

6. PRC Information Sciences Co.: Unpublished notes and tutorial material, January 1978.

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

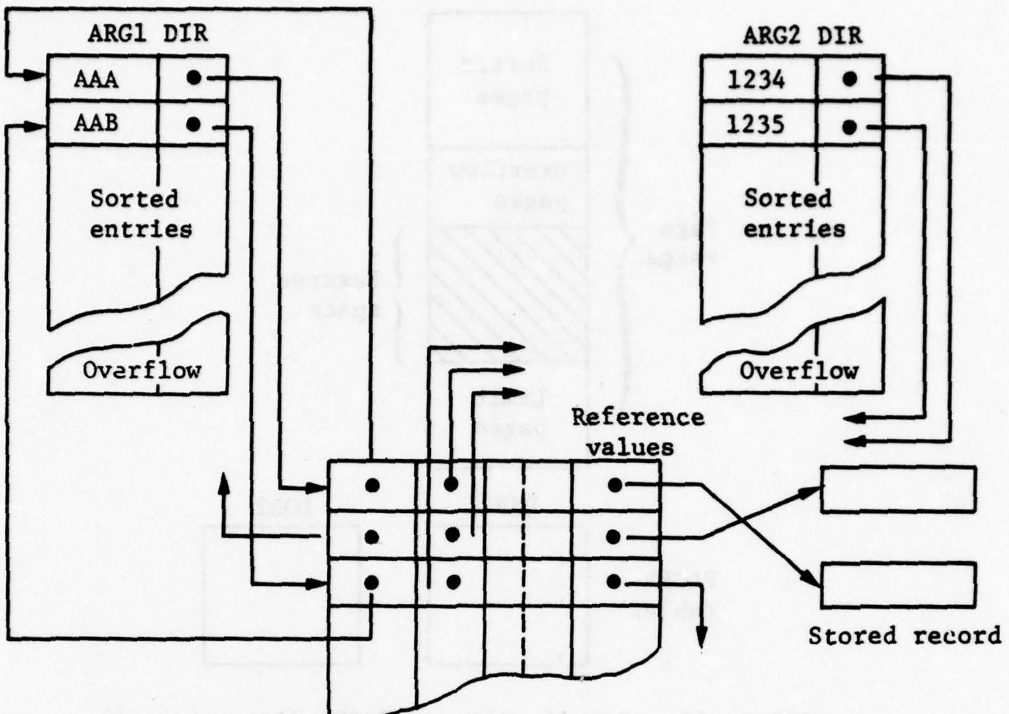


Figure 12. Argument/master directory organization.

directory. An entry in the master directory points to the corresponding record and back to related entries in the argument directories. (There are other distinctions between working directories and argument directories, mainly in how they are established and maintained. These differences are not of immediate concern in this study.)

In modeling this system, we have several alternatives. Our choice depends on the degree of fidelity desired. Figure 13 is a sketch of the simplest model. It represents the master directory as an RSG with selection criteria based on the combination of all attributes of interest.

This model produces a satisfactory set of paths based on qualification of any one of the directory attributes. Without the encoding level and introduction of "presursor sets,"[2] it does not represent the interdirectory searches possible in the real system.

To illustrate that we have some flexibility in moving features "upward" from the encoding level to the access-path level, we considered another model (Fig. 14), which is the one included in the executed demonstration. In this

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

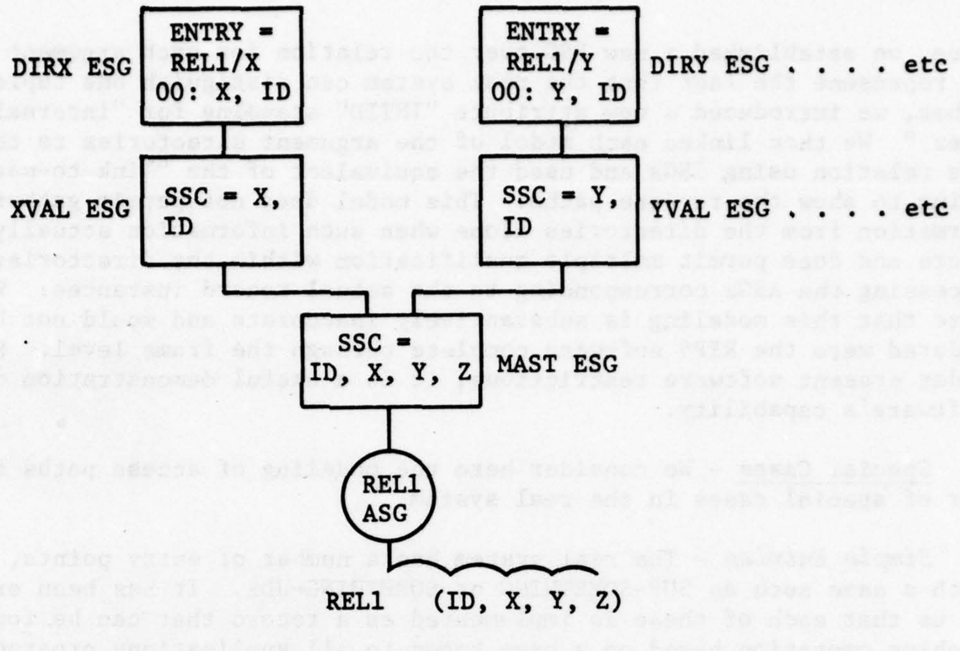


Figure 13. Simple modeling of argument/master directory.

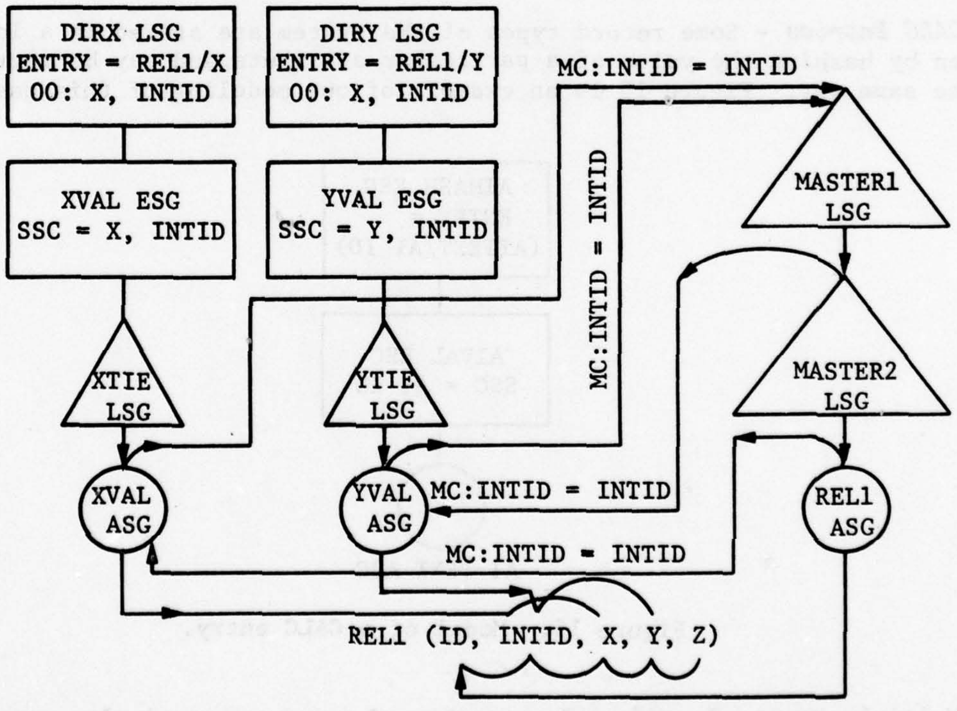


Figure 14. More elaborate modeling of argument/master directory.

case, we established a new PSG over the relation for each argument directory. To represent the fact that the real system can distiguish one tuple from another, we introduced a new attribute "INTID" standing for "internal identifier." We then linked each model of the argument directories to the PSG of the relation using JSGs and used the equivalent of the "link-to-master" modeling to show the reverse paths. This model does not permit gathering information from the directories alone when such information actually exists there and does permit multiple qualification within the directories before accessing the ASGs corresponding to the actual record instances. We recognize that this modeling is substantively inaccurate and would not be considered were the RIPS software complete through the frame level. However, under present software restrictions, it is a useful demonstration of the software's capability.

<u>Special Cases</u> - We consider here the modeling of access paths for a number of special cases in the real system.

Simple Entries - The real system has a number of entry points, often with a name such as SUP-SOMETHING or SOMETHING-HDR. It has been explained to us that each of these is implemented as a record that can be found via a hashing operation based on a name known to all applications programmers. For our purposes, we assume that the applications program has already executed the hashing algorithm and has available the address of the entry point. We therefore consider the point as an RSG with "ENTRY = DIRECT."

CALC Entries - Some record types of the system are stored in a location chosen by hashing the value of a particular attribute and may be retrieved in the same way. Figure 15 is an example of our modeling of this case.

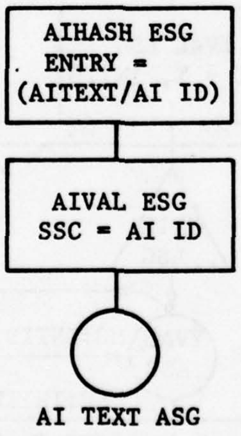


Figure 15. Model of a CALC entry.

Multiple Master Records - In a number of cases, a particular record type is designated as a detail of more that one chain originating at multiple master record types. This case is readily modeled. Each path is represented by a RSG over the PSG; each path may have different selection and ordering criteria and may be linked to various other relations corresponding to the master record types.

Multiple Detail Records - The PACER system contains cases in which a number of different record types are attached to the same chain. Our modeling recognizes the *logical* relationship in that each detail type is accessible from the master by establishing independent strings for each detail case. At the access-path level, the sequence of access operations and predicted number of accesses will not agree with those of the actual system. We regard the problem primarily as one of the encoding level and are examining the implications to parameters at that level. We note in passing that this implementation technique is an example of "clever" programming whose correspondence to the formalisms of any canonic approach is not yet well understood, but this topic is the subject of continued study.

Resulting Model

Scope of the Model - The resulting access path model for study and experimentation includes a sampling of the various techniques used in the PACER system. We have avoided exhaustive duplication. For example, we have included models of three argurment directories instead of the nine that actually exist. We modeled one working directory as a sample of that type of access. We have shown most of the paths corresponding to the forward direction of the chains among the record types modeled. We included some but not all of the corresponding "linked-master" paths, and most of the direct and hash entries for the records we modeled.

Presentation of the Model - Figure 16 is a sketch of the string path diagram. A sample string definition is included here as Table 3.

Implications

What Can Be Readily Modeled? - In this section, we have shown that the following access paths and techniques are readily modeled:

1) Single and multiple accesses to attributes within a relation;

2) Connection of instances of a single record type (modeled as a relation with optional ordering);

3) Subsetting of instances of a single record type based on values of a particular attribute;

4) Master/detail paths in the original system. These can be modeled in the master-to-detail direction and the linked-master direction. A detail type with more than one master type can be accurately modeled. A master type with more than one detail type can be modeled logically;

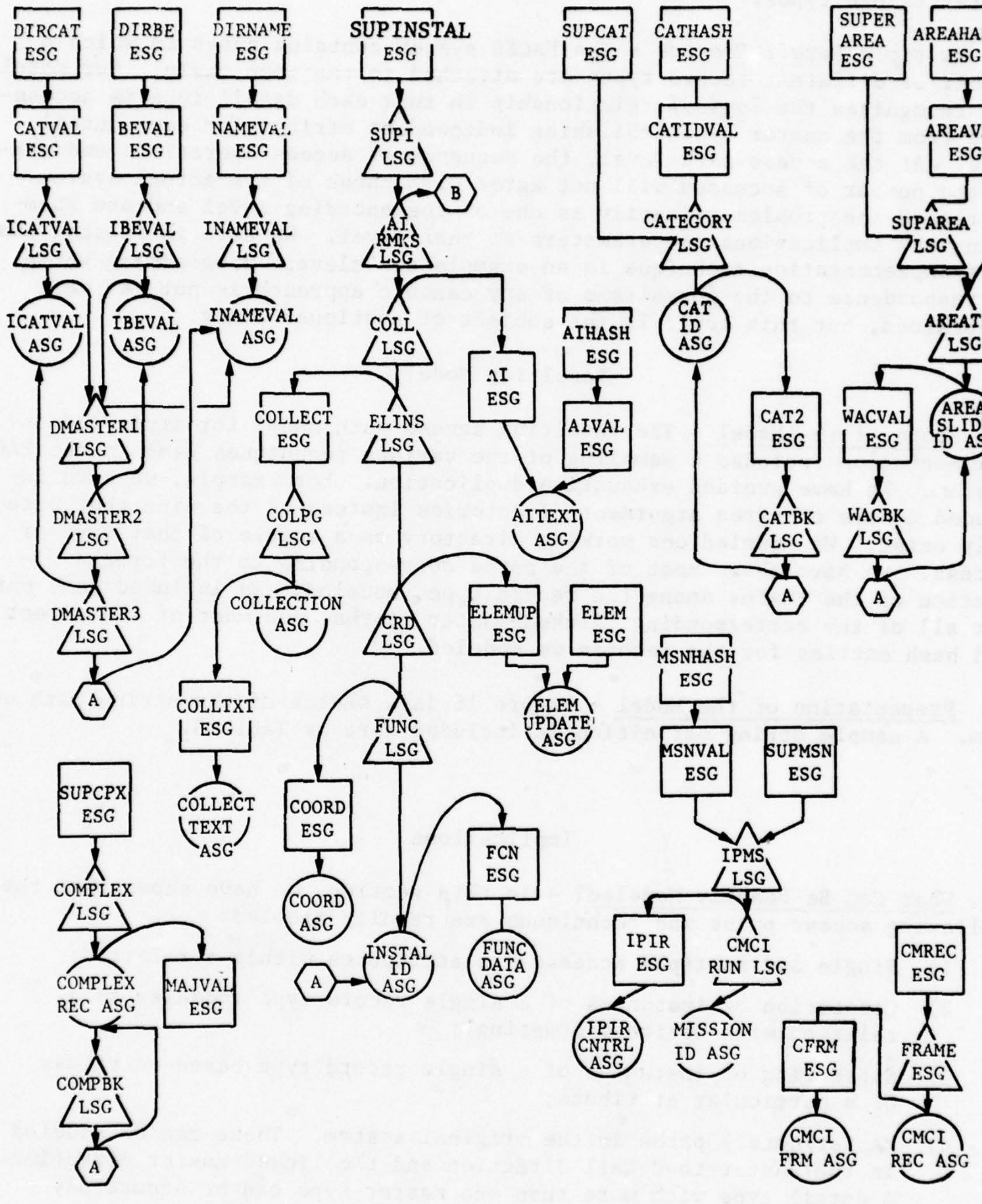


Figure 16. String path diagram of modeled PACER access paths.

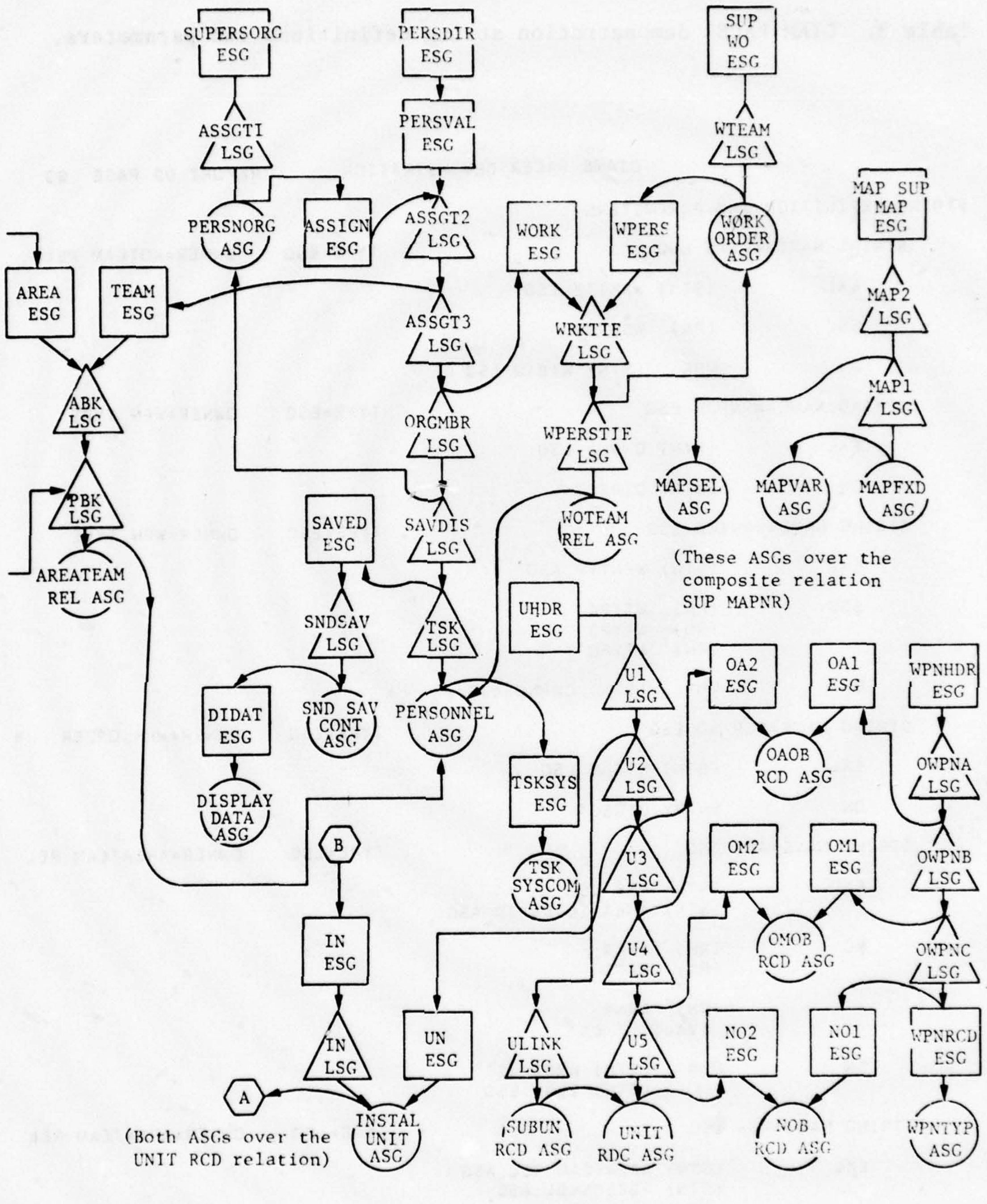


Figure 16. (concl)

Table 3. DIAM PACER demonstration string definitions and parameters.

```
                                   DIAMS PACER DEMONSTRATION       REPORT 02 PAGE  99
STRING DEFINITION AND PARAMETERS
     STRING NAME=WPERS ESG                                    TYPE=ESG     OWNER=WOTEAM REL
          EXL          (STN) WRKTIE LSG
          SSC          (RN)  WONRNR
          ON           MBR   (STN) WTEAM LSG
     STRING NAME=WPNHDR ESG                                   TYPE=ESG     OWNER=WPN TYPE
          EXL          (STN) OWPNA LSG
          ON           ENTRY DIRECT
     STRING NAME=WPNRCD ESG                                   TYPE=ESG     OWNER=WPN TYPE
          EXL          (STN) WPNTYP ASG
          SSC          (RN)  WTYPA
                       (RN)  WTYPB
                       (RN)  WTYPC
          ON           MBR   (STN) OWPNC LSG
     STRING NAME=SUP WO ESG                                   TYPE=ESG     OWNER=WRK ORDER
          EXL          (STN) WTEAM LSG
          ON           ENTRY DIRECT
     STRING NAME=ABK LSG                                      TYPE=LSG     OWNER=AREATEAM REL
          EXL          (STN) PBK LSG
                       (STN) AREA SLIDE ID ASG
          MC           (RN)  ARNR
                       (RN)  WACSH
                       (RN)  ARNR
                        (VAL)     0%
          ON           MBR   (STN) AREA ESG
                             (STN) TEAM ESG
     STRING NAME=PBK LSG                                      TYPE=LSG     OWNER=AREATEAM REL
          EXL          (STN) AREATEAM REL ASG
                       (STN) PERSONNEL ASG
          MC           (RN)  ANAME
                       (RN)  PNAME
                       (RN)  ANAME
                       *UNDF*
```

5) Entry points, both those generally accessible and those dependent on a function of a particular attribute;

6) Working and argument/master directories in terms of the main sorted tables.

What Cannot Be Readily Modeled as Access Paths? - The Relational DIAM methodology provides three types of strings as the primatives for modeling access paths. An implementation such as PACER can use encoding techniques whose correspondence to these primatives is not yet well understood.

Examples of such features include:

1) Multiple detail record types on a single chain to a master type. The difficulty is not in showing a path from the master to each detail but in attempting to show that all detail types may be intermixed on a single chain. While we might redefine a string type to show such a connection, it appears that we would lose the underlying mathematical discipline based on relational operators. The topic remains under examination.

2) Differing access paths to the same relation where the choice is based on some attribute value, a time of arrival, an operator's choice, or some other criterion. (e.g., the sorted versus unsorted portions of a directory) In some cases, the path may be different and so represent the presence of paths in some instances and the complete absence of paths in others. We consider that modeling can eventually be accomplished by the technique of "Restriction-Distribution".[2]

3) Access methods based on the parameters of the address space rather than on the data structure. An example is the use of "limit pages" in the construction of directories. In Senko's DIAM methodology, such aspects of the address space are allocated to the encoding level. In Schneider's modification, they are treated recursively at the relational level.

Relevance of Lower Levels of DIAM - Relational DIAM methodolgy makes sharp distinctions between adjacent levels. The string path level describes *what associations are implemented.* The encoding level describes *how each association is implemented.* The Frame Model describes *where it is implemented* in terms of address space. The physical device level describes the allocation of address space to devices and the parameters of such devices.

Descriptions of existing implemetations seldom fall neatly into these categories. The system design has often folded many of these characteristics into a single feature or technique. In a modeling study, we attempted to unfold them once more and focus on the options available at each level. The present study has examined only the information and access-path levels.

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

Topics from the PACER system that would be appropriately studied at the encoding level include:

1) Actual methods of encoding new records;
2) Representation of the mechanics of pointers;
3) "Factoring" of values to model the fact that redundant data values have been omitted;
4) Methods of assignment to address space;
5) Recognition of access methods that employ knowledge of the address space assignments (e.g., limit pages);
6) Actual required retrieval as a result of choice of storage methods.

DEMONSTRATION VIA QUERY COMPILER SOFTWARE (TASK 4.1.4)

Introduction

Software Background - The software system used for this demonstration was developed by Martin Marietta's Database Research Group. The first elements were developed under contract to NASA as a simulator to permit comparison of various generalized data management systems for proposed applications.[9,10,11]

Under subsequent company-funded research, the underlying concepts and implemented system were extended to form the basis of a generalized query compiler applicable to distributed databases.

Purpose of Demonstration - The purpose of the demonstration was to show the potential application of the prototype software to distributed databases. It was intended to show that the software can analyze each query to find and tabulate all legitimate paths to the data, then to decompose the query into sets of access procedures that are semantically sufficient to construct the answer by independent interrogation of each pertinent database.

---

9. 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.).

10. 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.

11. 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.

Specification of the Model

Analyses that led to an understanding of the information level and access path level models were described under Relational Model and Access Path Model. The remaining step was to encode those understandings in a form acceptable to the software system; such encoding is laborious but not particularly difficult. The input processor of the software contains extensive diagnostic tests to detect and report errors in syntax and consistency that might appear in the input data. These diagnostics did indeed detect numerous errors in the preliminary input data; the errors were subsequently corrected.

Input coding formats were originally designed for 80-column punched cards; in the present case, the data were input via computer terminal. The quantity of data is approximately equivalent to 2200 cards.

Specification of Traffic Load

Purpose and Nature of Traffic Load - An original purpose of the software system was to permit comparisons of data management systems and their various options. The comparisons were based on a common traffic load applied to each candidate system. The software contains very flexible capabilities for creating and/or using transaction streams that in general may include retrievals, additions, deletions, and changes.

Options Available - The software permits specification of sources, transactions, and a profile between them. Parameters specify rates, quantities, and durations of activity.

A transaction is an inquiry or update operation directed to the database. The software permits these to be fully or partially specified. Fully specified queries are declared in full as part of the input data (Fig. 4 is a sample). Partially specified queries have some or all of their details omitted originally; the missing details are supplied at execution time by query-generation routines that use the statistical parameters that make up part of the relational model specification.

Our present demonstration makes use of fully specified queries only.

Choice of Queries - In this study, the main purpose of the queries was to exercise the system and demonstrate the various relationships and access paths. In view of a query compiler to interrogate distributed databases, in this demonstration these consist of retrievals only.

Our general approach was to direct at least one query to each declared relation. We also tried to use an assortment of qualifiers, including some that are identifiers and some that are not. We also wished to include a few multistatement queries that interrogate different relations concurrently.

List of Queries - The list of queries actually executed appears as Report Number 08 in the computer output reproduced here as Table 4.

Results of Execution

Form of Results - Reports generated during execution of the queries include:

Report 01	Header page;
Report 02	Translation of input data reflecting the stored version;
Report 08	List of queries executed;
Report 12	Representation-dependent accessing language (RDAL) tabulation;
Report 22	Path tabulations for each query;
Report 23	Specification of the relations, their attributes, and indentifiers.

Reports 08, 23, and a sample of Report 12 appear in this report.

Evaluation of Results

Our main interest was in seeing that the compiler actually generated the proper paths to the desired data elements. Recall that the query compiler tabulates *all* legitimate routes to the data; these should include those that are obvious and others that are not so obvious. The routes may include some that are outrageously long or complicated; the compiler makes no judgment on accepting or rejecting any path.

Path Tabulation - We now turn our attention to the tabulations of Report 22 and comment on the results of each query. Table 5 is a sample page for Report 22. It is useful to refer to the string path diagram in Figure 16 at the same time.

Query No. 1: Illustrates alternative paths to a detail record via either of two master record types. The query compiler lists *all* legitimate paths.

Query No. 2: Illustrates five paths to a record; three different entry points are recognized. (The PERSDIR entry point was not applicable in this case because it can be used only with a qualifier of PNAME.)

Query No. 3: Illustrates alternative paths to a record using either a general entry point or a hashing function on the record. Because the qualifier specifies one value of an identifier, a single instance is found to qualify.

Query No. 4: Illustrates the accessing of a "many-to-many relation" via either of its associated types.

Table 4. DIAM PACER query tabulation.

```
                                      DIAMS PACER DEMONSTRATION                 REPORT 08 PAGE   1
QUERY TABULATION

QUERY NUMBER =     1      TRANSACTION NUMBERS =    1 THRU    1    AT TIME =      6.605
    GET  W3     .OF. OAOB RCD      /  *UTYP1   .WHERE.   UUNIT    =    00001

QUERY NUMBER =     2      TRANSACTION NUMBERS =    2 THRU    2    AT TIME =      6.960
    GET  P1     .OF. PERSONNEL     /  *PNAME   .WHERE.   ORGNR    =    00002

QUERY NUMBER =     3      TRANSACTION NUMBERS =    3 THRU    3    AT TIME =      7.707
    GET  MS2    .OF. MISSION ID    /  *FDATE   .WHERE.   MSNNR    =    00001

QUERY NUMBER =     4      TRANSACTION NUMBERS =    4 THRU    4    AT TIME =      8.255
    GET  P7     .OF. WOTEAM REL    /  *WNAME   .WHERE.   WONRNR   =    00001

QUERY NUMBER =     5      TRANSACTION NUMBERS =    5 THRU    6    AT TIME =     10.030
    GET  C3A    .OF. CAT ID  /    CATEG   .WHERE.   UGRP     =   00001
    GET  C3B    .OF. INSTAL ID     /  *NAME    .WHERE.   CAT      =    C3A     /   CATEG

QUERY NUMBER =     6      TRANSACTION NUMBERS =    7 THRU    7    AT TIME =     10.178
    GET  AIQ1   .OF. AITEXT  /  *AI ID   .WHERE.   CDATE   =   00001

QUERY NUMBER =     7      TRANSACTION NUMBERS =    8 THRU    8    AT TIME =     13.694
    GET  U2     .OF. INSTAL UNIT   /  *UI INS  /  *UI UNIT        .WHERE.   UI UNIT        =     00001

QUERY NUMBER =     8      TRANSACTION NUMBERS =    9 THRU    9    AT TIME =     16.243
    GET  U4     .OF. UNIT RCD      /  *UID     .WHERE.   UMOD     =    00001

QUERY NUMBER =     9      TRANSACTION NUMBERS =   10 THRU   10    AT TIME =     22.472
    GET  U5     .OF. UNIT RCD      /  *UID     /  *UMOD     .WHERE.   SUBUMOD        =    00001

QUERY NUMBER =    10      TRANSACTION NUMBERS =   11 THRU   11    AT TIME =     24.497
    GET  CLQ2   .OF. COLLECT TEXT  /  *CTXT    .WHERE.   COLNR    =    00001

QUERY NUMBER =    11      TRANSACTION NUMBERS =   12 THRU   12    AT TIME =     25.079
    GET  ARQ1   .OF. AREA SLIDE ID       /  *CSTAT   .WHERE.   WACSH   =   00001

QUERY NUMBER =    12      TRANSACTION NUMBERS =   13 THRU   15    AT TIME =     25.444
    GET  U3A    .OF. INSTAL ID     /  NAME     .WHERE.   CAT      =    00001
    GET  U3B    .OF. INSTAL UNIT   /  UI UNIT           .WHERE.   UI INS   =   U3A     /   NAME
    GET  U3C    .OF. UNIT RCD      /  *UNAM    /  *SUBUMOD        .WHERE.   UID      =   U3B     /   UI UNIT

QUERY NUMBER =    13      TRANSACTION NUMBERS =   16 THRU   16    AT TIME =     30.581
    GET  M2     .OF. SUP MAPNR     /  *MAPID   .WHERE.   SCALE    =    00001

QUERY NUMBER =    14      TRANSACTION NUMBERS =   17 THRU   17    AT TIME =     32.443
    GET  MS1    .OF. MISSION ID    /  *MSNNR   .WHERE.   FDATE    =    00001

QUERY NUMBER =    15      TRANSACTION NUMBERS =   18 THRU   20    AT TIME =     33.442
    GET  P3A    .OF. AREATEAM REL  /  ANAME    .WHERE.   ARNR     =    00002
    GET  P3B    .OF. PERSONNEL     /  ORGNR    .WHERE.   PNAME    =    P3A     /   ANAME
    GET  P3C    .OF. PERSNORG      /  *ORG     /  *PRNTR   .WHERE.   ORG      =    P3B     /   ORGNR

QUERY NUMBER =    16      TRANSACTION NUMBERS =   21 THRU   21    AT TIME =     33.494
    GET  C1     .OF. CAT ID  /  *NTR      .WHERE.   CATEG   =   00001

QUERY NUMBER =    17      TRANSACTION NUMBERS =   22 THRU   22    AT TIME =     33.731
    GET  CPQ1   .OF. COMPLEX REC   /  *CPXTR   .WHERE.   CMAJR    =    00001
```

Table 4. (cont).

```
                                              DIAMS PACER DEMONSTRATION          REPORT 08 PAGE   2
QUERY TABULATION

QUERY NUMBER =     18      TRANSACTION NUMBERS =   23 THRU     23     AT TIME =    35.228
     GET  P4     .OF. PERSNORG       /  *ORG      /  *PRNTR    .WHERE.   PRNTR   =    00002

QUERY NUMBER =     19      TRANSACTION NUMBERS =   24 THRU     24     AT TIME =    41.668
     GET  C2     .OF. CAT ID  /  *CATEG   .WHERE.   CRIT     =    00001

QUERY NUMBER =     20      TRANSACTION NUMBERS =   25 THRU     25     AT TIME =    42.767
     GET  FNQ1   .OF. FUNCTION DATA          /  *RMKS     .WHERE.   INS      =    00001

QUERY NUMBER =     21      TRANSACTION NUMBERS =   26 THRU     26     AT TIME =    44.527
     GET  P8     .OF. WRK ORDER      /  *WONR     .WHERE.   DUEDATE       =    00001

QUERY NUMBER =     22      TRANSACTION NUMBERS =   27 THRU     27     AT TIME =    44.581
     GET  MS5    .OF. CMCI FRM       /  *HGT      .WHERE.   CMFRM    =    00001

QUERY NUMBER =     23      TRANSACTION NUMBERS =   28 THRU     29     AT TIME =    49.654
     GET  S3A    .OF. SND SAV CONT  /   SDID      .WHERE.   RCVR     =    00001
     GET  S3B    .OF. DISPLAY DATA  /  *DISDAT    .WHERE.   SDID2    =    S3A      /   SDID

QUERY NUMBER =     24      TRANSACTION NUMBERS =   30 THRU     30     AT TIMI =    49.717
     GET  ELQ1   .OF. ELEM UPDATE    /  *INS      .WHERE.   PDATE    =    00001

QUERY NUMBER =     25      TRANSACTION NUMBERS =   31 THRU     31     AT TIME =    59.170
     GET  W2     .OF. WPN TYPE       /  *WNAM     .WHERE.   WOTHER   =    00001

QUERY NUMBER =     26      TRANSACTION NUMBERS =   32 THRU     32     AT TIME =    72.318
     GET  MS4    .OF. CMCI REC       /  *CNRUN    .WHERE.   MSNNO    =    00001

QUERY NUMBER =     27      TRANSACTION NUMBERS =   33 THRU     33     AT TIME =    80.767
     GET  W1     .OF. WPN TYPE       /  *WOTHER   .WHERE.   WNAM     =    00002

QUERY NUMBER =     28      TRANSACTION NUMBERS =   34 THRU     34     AT TIME =    88.750
     GET  S2     .OF. DISPLAY DATA  /  *SDID2     .WHERE.   DISIZ    =    00002

QUERY NUMBER =     29      TRANSACTION NUMBERS =   35 THRU     35     AT TIME =   102.507
     GET  P5     .OF. AREATEAM REL  /  *ARNR      /  *ANAME    .WHERE.   ANAME    =    00002

QUERY NUMBER =     30      TRANSACTION NUMBERS =   36 THRU     36     AT TIME =   106.994
     GET  M1     .OF. SUP MAPNR      /  *MAPTOP   .WHLRE.   MAPID    =    00001

QUERY NUMBER =     31      TRANSACTION NUMBERS =   37 THRU     37     AT TIME =   110.845
     GET  CLQ1   .OF. COLLECTION     /  *LENG     .WHERE.   COLID    =    00001

QUERY NUMBER =     32      TRANSACTION NUMBERS =   38 THRU     38     AT TIME =   117.358
     GET  M4     .OF. SUP MAPNR      /  *MAPID    .WHERE.   MAPTOP   =    00001 .AND.SCALE     =     00001

QUERY NUMBER =     33      TRANSACTION NUMBERS =   39 THRU     39     AT TIME =   136.197
     GET  S1     .OF. DISPLAY DATA  /  *DISDAT    .WHERE.   SDID2    =    00001

QUERY NUMBER =     34      TRANSACTION NUMBERS =   40 THRU     41     AT TIME =   139.481
     GET  P2A    .OF. PERSONNEL      /  *ORGNR    .WHERE.   RANK     =    00001
     GET  P2B    .OF. PERSNORG       /  *PRNTR    .WHERE.   ORG      =    P2A      /   ORGNR
```

Table 4. (concl).

```
                                        DIAMS PACER DEMONSTRATION                  REPORT 08 PAGE   3

  QUERY TABULATION

QUERY NUMBER =     35      TRANSACTION NUMBERS =    42 THRU     42      AT TIME =    143.641
    GET  M3      .OF. SUP MAPNR      /  *MAPID   .WHERE.   LINE3    =    00001

QUERY NUMBER =     36      TRANSACTION NUMBERS =    43 THRU     43      AT TIME =    148.882
    GET  P6      .OF. TSK SYS COM    /  *REF     .WHERE.   PNAME2   =    00001

QUERY NUMBER =     37      TRANSACTION NUMBERS =    44 THRU     44      AT TIME =    185.641
    GET  D4      .OF. INSTAL ID      /  *NAME    .WHERE.   MAJ      =    00003

QUERY NUMBER =     38      TRANSACTION NUMBERS =    45 THRU     45      AT TIME =    188.012
    GET  W5      .OF. NOB RCD        /  *NOID    .WHERE.   NOWTYP   =    000C1

QUERY NUMBER =     39      TRANSACTION NUMBERS =    46 THRU     46      AT TIME =    199.839
    GET  MS3     .OF. IPIR CNTRL     /  *FUN H   .WHERE.   MSNN     =    00001

QUERY NUMBER =     40      TRANSACTION NUMBERS =    47 THRU     47      AT TIME =    216.230
    GET  M5      .OF. SUP MAPNR      /  *MAPID   /  *MAPTOP  /  *SCALE   .WHERE.   LINE3    =    00010

QUERY NUMBER =     41      TRANSACTION NUMBERS =    48 THRU     48      AT TIME =    242.518
    GET  W4      .OF. OMOB RCD       /  *CMCAT   .WHERE.   OMID     =    00001

QUERY NUMBER =     42      TRANSACTION NUMBERS =    49 THRU     49      AT TIME =    258.438
    GET  U1      .OF. INSTAL UNIT    /  *UI INS  /  *UI UNIT       .WHERE.   UI INS   =    00001

QUERY NUMBER =     43      TRANSACTION NUMBERS =    50 THRU     50      AT TIME =    266.356
    GET  CRQ1    .OF. COORDINATE     /  *MTD DATE        .WHERE.   PRIORITY       =    00001

QUERY NUMBER =     44      TRANSACTION NUMBERS =    51 THRU     51      AT TIME =    267.737
    GET  D3      .OF. INSTAL ID      /  *MAJ     .WHERE.   CAT      =    00001 .AND.BE        =    00002

QUERY NUMBER =     45      TRANSACTION NUMBERS =    52 THRU     52      AT TIME =    355.876
    GET  D2      .OF. INSTAL ID      /  *BE      .WHERE.   CAT      =    00002

QUERY NUMBER =     46      TRANSACTION NUMBERS =    53 THRU     53      AT TIME =    457.474
    GET  D1      .OF. INSTAL ID      /  *CAT     .WHERE.   NAME     =    00002
```

Table 5. DIAM PACER path expansion (sample).

```
                                   DIAMS PACER DEMONSTRATION                               REPORT 22
  STRBM PATH STATS - QUERY 00001

QUERY NUMBER =      1      TRANSACTION NUMBERS =      1 THRU      1      AT TIME =    457.474
   GET  D1      .OF. INSTAL ID      /   *CAT      .WHERE.   NAME      =   00002

   ORDER NO. =    1   -- D1          .
      SET NAME =     D1
         CONJ NO. =        1

         *** TARGET LIST ***   (DSN NAME = INSTAL ID                               )

         NAME                              =     2
         CAT

              *** COMPOSITION ***

              INSTAL ID ASG

         PATH    1   STRING OR RN                                            INSTANCES PER PREVIOUS STRING
                                                                     PRESENT          QUALIFIED          TRAVERSED
                     (STN)        SUPINSTAL ESG                    1.0000000          1.0000000          1.0000000
                     (STN)        SUPI LSG                     80000.0000         80000.0000         80000.0000
                     (STN)        AI RMKS LSG                      1.0000000          1.0000000          1.0000000
                     (STN)        COLL LSG                         1.0000000          1.0000000          1.0000000
                     (STN)        ELINS LSG                        1.0000000          1.0000000          1.0000000
                     (STN)        CRD LSG                          1.0000000          1.0000000          1.0000000
                     (STN)        FUNC LSG                         1.0000000          1.0000000          1.0000000
                     (STN)        INSTAL ID ASG                    1.0000000          0.0000250          0.6666666
                      (RN)        NAME                             1.0000000          1.0000000          1.0000000
                      (RN)        BE                               1.0000000          1.0000000          1.0000000
                      (RN)        BESUF                            1.0000000          1.0000000          1.0000000
                      (RN)        MAJ                              1.0000000          1.0000000          1.0000000
                      (RN)        MIN                              1.0000000          1.0000000          1.0000000
                      (RN)        CAT                              1.0000000          1.0000000          1.0000000
                           (CARDINALITY OF THIS PATH =     533334)
                           (NUMBER OF TUPLES QUALIFIED =          2.00)
         PATH    2   STRING OR RN                                            INSTANCES PER PREVIOUS STRING
                                                                     PRESENT          QUALIFIED          TRAVERSED
                     (STN)        DIRNAME ESG                      1.0000000          1.0000000          1.0000000
                     (STN)        NAME VAL ESG                 80000.0000             2.0000000      53333.8281
                     (STN)        INAMEVAL LSG                     1.0000000          1.0000000          1.0000000
                     (STN)        INAMEVAL ASG                     1.0000000          1.0000000          1.0000000
                     (STN)        DMASTER1 LSG                     1.0000000          1.0000000          1.0000000
                     (STN)        DMASTER2 LSG                     1.0000000          1.0000000          1.0000000
                     (STN)        DMASTER3 LSG                     1.0000000          1.0000000          1.0000000
                     (STN)        INSTAL ID ASG                    1.0000000          1.0000000          1.0000000
                      (RN)        NAME                             1.0000000          1.0000000          1.0000000
                      (RN)        BE                               1.0000000          1.0000000          1.0000000
                      (RN)        BESUF                            1.0000000          1.0000000          1.0000000
                      (RN)        MAJ                              1.0000000          1.0000000          1.0000000
                      (RN)        MIN                              1.0000000          1.0000000          1.0000000
                      (RN)        CAT                              1.0000000          1.0000000          1.0000000
                           (CARDINALITY OF THIS PATH =     373338)
                           (NUMBER OF TUPLES QUALIFIED =          2.00)
```

Query No. 5:	This is the first multiargument query in which two different relations are referenced in the query. The query statements in Report 22 reflect the process of "balancing" to provide a symmetrical statement of the retrieval; in this case, the term CATEG = C3B/CAT has been appended to the original query as it appears in Table 4.[2] Possible paths for each argument are tabulated separately, then the possible consolidations are tabulated as shown beginning on page 24 of Report 22. In this query, we observe some differences in the values obtained for the number of tuples qualified (e.g., Path 1 versus Path 2). We have determined that this difference results from insufficient precision in the numerical values included in the matching criteria of some L-strings; the parameter is input as a two-digit number representing percentage. Correcting this will eventually require minor changes to several program routines.
Query No. 6:	This query illustrates access that is qualified upon an attribute that is neither an identifier nor included in any directory or hashing mechanism.
Query No. 7:	Illustrates the accessing of a "many-to-many relation" via either of its associated types.
Queries No. 8 & 9:	Illustrate cases in which the underlying relation is accessed by more than one A-string. In No. 8, the object attribute and qualifier attribute are within the same A-string. In No. 9, they are from different A-strings.
Query No. 10:	Illustrates various paths to a record using either a general entry point or a hashing function.
Query No. 12:	Illustrates the enormous complexity that can exist in the access paths; the printout requires 177 pages. This is a three-statement query in which the user visualized obtaining a set of NAMES qualified on CAT, then obtaining UI UNIT based on the first set, then obtaining attributes of UNIT RCD based on the second set. The balancing operation of the query compiler recognized that there were alternative ways of stating the query and appended the balancing terms shown in the query statements as they appear in Report 22. The resulting alternative paths for each statement are then shown followed by the possible consolidations. We observe here the effects if an identified program logic error. The number of tuples qualified differs between some paths (e.g., Path 25 versus Path 26). We have determined

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

that the "same string" logic incorporated in Path 26 is in error in that a qualification restriction has been used twice; there is no conceptual difficulty in correcting this logic.

Query No. 13:	Illustrates the accessing of a relation covered by more than one A-string.
Query No. 14:	Illustrates one path to a detail record. Compare with Query No. 3.
Query No. 15:	Illustrates another three-statement query.
Query No. 16:	Illustrates access based on an identifier with alternative paths via general entry and via hashing.
Queries No. 17, 18, & 19:	Illustrate a direct route to the desired data and a less obvious route via linked relations.
Query No. 20:	Illustrates a single path to a detailed record that is quite deep in the hierarchy.
Query No. 21:	Illustrates paths to a record both directly and via intermediate relations.
Query No. 22:	Illustrates a path to a subordinate record via successive master types.
Query No. 23:	Illustrates a two-statement query.
Query No. 24:	Illustrates the use of a direct entry and entry via a connected master record.
Queries No. 25 & 26:	Illustrate access based on a nonidentifier.
Query No. 27:	Illustrates retrieval based on two values of an identifier.
Query No. 28:	Illustrates access based on a nonidentifier.
Query No. 29:	Illustrates accessing a "many-to-many relation" via either of its associated types.
Query No. 30:	Illustrates access of a relation covered by more than one A-string. Compare with Query No. 13.
Query No. 31:	Illustrates a single path based on an identifier.
Query No. 32:	Illustrates access based on two qualifiers that are covered by different A-strings.
Query No. 33:	Illustrates access based on a qualifier that is a portion of an identifier.
Query No. 34:	Illustrates another two-statement query involving a detail type and its master type.
Query No. 35:	Illustrates access of a relation covered by more than one A-string.
Query No. 36:	Illustrates access to a detail type based on a qualifier that matches an identifier of a master type.

Query No. 37:	Illustrates access based on a qualifier that is not an identifier and is not included in any directory or hashing selection.
Query No. 38:	Illustrates alternative paths to a detail record via either of two master record types.
Query No. 39:	Illustrates retrieval involving one level of chaining.
Query No. 40:	Illustrates access to a relation covered by more than one A-string. In this case, the two qualifiers are under different A-strings.
Query No. 41:	Illustrates alternative paths to a detail record via either of two master record types.
Query No. 42:	Illustrates accessing a "many-to-many relation" via either of its associated types.
Query No. 43:	Illustrates retrieval involving one level of chaining.
Query No. 44:	Illustrates retrieval based on two qualifiers, both of which are implemented in directories. This query also results in many permitted paths, mainly because there are many possible routes to the INSTAL ID relation.
Query No. 45:	Illustrates retrieval of an object attribute based on a qualifier attribute where both object and qualifier are implemented in directories. Note Paths 27, 28, 29, and 30 in which the response to a query is completely obtained from the directories.
Query No. 46:	Again illustrates retrieval where both object and qualifier are available in directories; in this case, the qualifier is also an identifier. Paths 27 and 28 show the query resolved using only the directories.

Interpretation of Cardinality - The value of cardinality computed for each path is the expected value for the total number of individual instances of attribute values and string instances to be retrieved. In the most pessimistic case (every attribute and every pointer stored independently and randomly), the number of accesses would equal the cardinality. In any practical case, many values are stored by contiguity with one another, and many pointers are stored with data; the result is that fewer accesses are actually required. Considerations of storage methods are examined in the *encoding level* of DIAM, which was not included in this study.

Therefore, cardinality values may be examined for rough comparisons of retrieval effort but should not be taken as a detailed figure of merit.

Representation-Dependent Accessing Language (RDAL) - RDAL consists of sets of detailed procedural steps that accomplish the accesses specified by a query. It corresponds to the Data Manipulation Language (DML) of most data management systems.

The access-path level of the software generates RDAL statements as an output to lower levels of the software model. RDAL syntax used for simulation resembles various DMLs but is not intended to duplicate any particular one. If the software were used as a query compiler to interface to a specific GDMS, then the RDAL would be translated to the syntax of the specified DML.

To illustrate the nature of the RDAL produced by the simulation software, we include the results of RDAL (Report 12) for a single query (Table 6). A complete set of RDAL would be too voluminous and repetitious to be of interest. Statements of the nature "DO LAST RDAL nnn TIMES" correspond to the loops that might be written by a programmer in his host language. The number of iterations shown in the RDAL is a statistical estimate for the number of iterations that would result in the actual case.

Timing (Cost) of Execution - In considering the potential value of prototype software to analysis of queries for distributed systems, an important consideration is the time of analysis compared to the time otherwise needed for response.

We examined the actual timing of our batch executions of software to obtain estimates of the CPU time devoted to analysis of the queries. Our two final runs were:

Date	No. of Queries	CPU Time, sec
3/31/78	1	29.9
4/13/78	46	48.4

If we allocate the difference in time between the two cases to the 45 additional queries executed, then we infer that the average processing time per query is about 0.4 second. Based on these and earlier runs, we have estimated that accepting and processing the model's input data take about 20 seconds.

We concluded that the 0.4-second average analysis time is an acceptable increment to be added to typical times associated with retrieval from large files stored on disks; for many existing database management systems (DBMSs), such retrievals are usually considered to require from 1 to 3 seconds.

Table 6. DIAM PACER demonstration RDAL for a single query (Report 12).

```
                                        DIAMS PACER DEMONSTRATION                               REPORT 12 PAGE    1
RDAL TABULATION

QUERY NUMBER =      1

   PATH NUMBER =      1
   ;GET SUPINSTAL ESG       ;GET NEXT =       1.0      .OF. SUPI LSG        ;GET AI RMKS LSG  ;GET COLL LSG      ;GET ELINS LSG
   ;GET CRD LSG       ;GET FUNC LSG       ;GET NEXT =       0.00002 .OF. INSTAL ID ASG          ;GET NAME   ;GET BE     ;GET BESUF
   ;GET MAJ     ;GET MIN     ;GET CAT      ;DO  LAST    00007 RDAL        0.0       TIMES ;GET NEXT =       0.66664 .OF. INSTAL ID AS
G     ;GET NAME    ;GET BE      ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL        0.0      TIMES
   ;DO  LAST    00022 RDAL  79999.0        TIMES

   PATH NUMBER =      2
   ;GET DIRNAME ESG   ;GET NEXT =       1.0         .OF. NAME VAL ESG ;GET NEXT =       1.0        .OF. INAMEVAL LSG ;GET INAMEVAL ASG
   ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET NEXT =       1.0       .OF. INSTAL ID ASG          ;GET NAME     ;GET
BE     ;GET BESUF   ;GET MAJ      ;GET MIN         ;GET CAT       ;DO  LAST    00007 RDAL        0.0       TIMES ;DO  LAST    00013 RDAL
     0.0     TIMES ;DO  LAST    00015 RDAL       1.0       TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG

   PATH NUMBER =      3
   ;GET SUPINSTAL ESG       ;GET NEXT =       1.0      .OF. SUPI LSG        ;GET AI RMKS LSG  ;GET IN ESG ;GET NEXT =       1.0
   .OF. IN LSG ;GET INSTAL UNIT ASG      ;GET NEXT =       0.0        .OF. INSTAL ID ASG          ;GET NAME     ;GET BE      ;GET BESUF
   ;GET MAJ     ;GET MIN     ;GET CAT      ;DO  LAST    00007 RDAL        0.0       TIMES ;GET NEXT =       0.13332 .OF. INSTAL ID AS
G     ;GET NAME    ;GET BE      ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL        0.0      TIMES
   ;DO  LAST    00018 RDAL        3.99976 TIMES ;DO  LAST    00022 RDAL  79999.0        TIMES

   PATH NUMBER =      4
   ;GET SUPCPX ESG    ;GET NEXT =       1.0      .OF. COMPLEX LSG   ;GET COMPLEX REC ASG        ;GET MAJVAL ESG   ;GET NEXT =       1
.0       .OF. COMPBK LSG   ;GET NEXT =       0.00002 .OF. INSTAL ID ASG         ;GET NAME     ;GET BE       ;GET BESUF   ;GET MAJ
   ;GET MIN     ;GET CAT      ;DO  LAST    00007 RDAL        0.0       TIMES ;GET NEXT =       0.66664 .OF. INSTAL ID ASG        ;GET
NAME    ;GET BE      ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL        0.0      TIMES ;DO  LAST
   00017 RDAL        9.0       TIMES ;DO  LAST    00021 RDAL    7999.0        TIMES

   PATH NUMBER =      5
   ;GET SUPER AREA ESG      ;GET NEXT =       1.0      .OF. SUPAREA LSG   ;GET AREATIE LSG   ;GET AREA SLIDE ID ASG   ;GET WACVAL
ESG    ;GET NEXT =       1.0      .OF. WACBK LSG     ;GET NEXT =       0.00002 .OF. INSTAL ID ASG          ;GET NAME     ;GET BE
   ;GET BESUF   ;GET MAJ      ;GET MIN     ;GET CAT      ;DO  LAST    00007 RDAL        0.0       TIMES ;GET NEXT =       0.66664 .OF.
INSTAL ID ASG        ;GET NAME     ;GET BE       ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL        0
.0     TIMES ;DO  LAST    00017 RDAL        7.0       TIMES ;DO  LAST    00022 RDAL    9999.0        TIMES

   PATH NUMBER =      6
   ;GET SUPCAT ESG    ;GET NEXT =       1.0      .OF. CATEGORY LSG ;GET CAT ID ASG     ;GET CAT2 ESG       ;GET NEXT =       1.0
   .OF. CATBK LSG    ;GET NEXT =       0.00002 .OF. INSTAL ID ASG        ;GET NAME     ;GET BE       ;GET BESUF   ;GET MAJ      ;GET
MIN     ;GET CAT     ;DO  LAST    00007 RDAL        0.0       TIMES ;GET NEXT =       0.66664 .OF. INSTAL ID ASG        ;GET NAME
   ;GET BE      ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL        0.0      TIMES ;DO  LAST    00017
   RDAL       39.0       TIMES ;DO  LAST    00021 RDAL    1999.0        TIMES

   PATH NUMBER =      7
   ;GET DIRNAME ESG   ;GET NEXT =       1.0         .OF. NAME VAL ESG ;GET NEXT =       1.0        .OF. INAMEVAL LSG ;GET INAMEVAL ASG
   ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET NEXT =       1.0       .OF. ICATVAL ASG   ;GET CAT     ;DO  LAST
   00002 RDAL        0.0       TIMES ;DO  LAST    00008 RDAL        0.0       TIMES ;DO  LAST    00010 RDAL        1.0      TIMES ;GET
NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPINSTAL ESG       ;GET NEXT =       1.0      .OF. SUPI LSG       ;GET AI RMKS LSG
```

Table 6. (cont).

```
                                           DIAMS PACER DEMONSTRATION                                REPORT 12 PAGE   2
RDAL TABULATION

   ;GET COLL LSG     ;GET ELINS LSG    ;GET CRD LSG     ;GET FUNC LSG      ;GET NEXT =      0.00002 .OF. INSTAL ID ASG
   ;GET NAME   ;GET BE       ;GET BESUF   ;GET MAJ      ;GET MIN     ;GET CAT     ;DO  LAST   00007 RDAL       0.0     TIMES ;GET
  NEXT =      0.66664 .OF. INSTAL ID ASG        ;GET NAME    ;GET BE      ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT     ;DO
  LAST    00007 RDAL       0.0      TIMES ;DO  LAST    00022 RDAL  79999.0       TIMES

   PATH NUMBER =      8
   ;GET DIRNAME ESG   ;GET NEXT =       1.0       .OF. NAME VAL ESG ;GET NEXT =       1.0       .OF. INAMEVAL LSG ;GET INAMEVAL ASG
   ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET NEXT =       1.0       .OF. ICATVAL ASG   ;GET CAT      ;DO  LAST
   00002 RDAL       0.0      TIMES ;GET   SAME    .OF. DMASTER3 LSG ;GET NEXT =       1.0       .OF. INSTAL ID ASG      ;GET NAME
   ;GET BE        ;GET BESUF   ;GET MAJ       ;GET MIN       ;GET CAT       ;DO  LAST    00007 RDAL       0.0      TIMES ;DO  LAST    0001
   RDAL       0.0      TIMES ;DO  LAST    00019 RDAL        1.0        TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG

   PATH NUMBER =      9
   ;GET DIRNAME ESG   ;GET NEXT =       1.0       .OF. NAME VAL ESG ;GET NEXT =       1.0       .OF. INAMEVAL LSG ;GET INAMEVAL ASG
   ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET NEXT =       1.0       .OF. ICATVAL ASG   ;GET CAT      ;DO  LAST
   00002 RDAL       0.0      TIMES ;DO  LAST    00008 RDAL       0.0      TIMES ;DO  LAST    00010 RDAL       1.0     TIMES ;GET
  NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPINSTAL ESG        ;GET NEXT =       1.0       .OF. SUPI LSG       ;GET AI RMKS LSG
   ;GET IN ESG ;GET NEXT =       1.0       .OF. IN LSG ;GET INSTAL UNIT ASG     ;GET NEXT =       0.0       .OF. INSTAL ID ASG
   ;GET NAME    ;GET BE       ;GET BESUF   ;GET MAJ       ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL       0.0     TIMES ;GET
  NEXT =      0.13332 .OF. INSTAL ID ASG        ;GET NAME    ;GET BE       ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT      ;DO
  LAST    00007 RDAL       0.0      TIMES ;DO  LAST    00018 RDAL       3.99976 TIMES ;DO  LAST    00022 RDAL   79999.0       TIMES

   PATH NUMBER =     10
   ;GET DIRNAME ESG   ;GET NEXT =       1.0       .OF. NAME VAL ESG ;GET NEXT =       1.0       .OF. INAMEVAL LSG ;GET INAMEVAL ASG
   ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET NEXT =       1.0       .OF. ICATVAL ASG   ;GET CAT      ;DO  LAST
   00002 RDAL       0.0      TIMES ;DO  LAST    00008 RDAL       0.0      TIMES ;DO  LAST    00010 RDAL       1.0    TIMES ;GET
  NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPCPX ESG     ;GET NEXT =       1.0       .OF. COMPLEX LSG   ;GET COMPLEX REC ASG
   ;GET MAJVAL ESG    ;GET NEXT =       1.0       .OF. COMPBK LSG     ;GET NEXT =        0.00002 .OF. INSTAL ID ASG          ;GET NAME
   ;GET BE       ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT       ;DO  LAST    00007 RDAL       0.0      TIMES ;GET NEXT =      0
  .66664 .OF. INSTAL ID ASG          ;GET NAME    ;GET BE       ;GET BESUF   ;GET MAJ       ;GET MIN      ;GET CAT     ;DO  LAST   00007
   RDAL       0.0      TIMES ;DO  LAST    00017 RDAL       9.0       TIMES ;DO  LAST    00021 RDAL    7999.0       TIMES

   PATH NUMBER =     11
   ;GET DIRNAME ESG   ;GET NEXT =       1.0       .OF. NAME VAL ESG ;GET NEXT =       1.0       .OF. INAMEVAL LSG ;GET INAMEVAL ASG
   ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET NEXT =       1.0       .OF. ICATVAL ASG   ;GET CAT      ;DO  LAST
   00002 RDAL       0.0      TIMES ;DO  LAST    00008 RDAL       0.0      TIMES ;DO  LAST    00010 RDAL       1.0     TIMES ;GET
  NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPER AREA ESG       ;GET NEXT =       1.0       .OF. SUPAREA LSG   ;GET AREATIE LSG
   ;GET AREA SLIDE ID ASG   ;GET WACVAL ESG      ;GET NEXT =       1.0       .OF. WACBK LSG       ;GET NEXT =       0.00002 .OF. INSTAL
   ID ASG       ;GET NAME     ;GET BE       ;GET BESUF   ;GET MAJ       ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL       0.0
   TIMES ;GET NEXT =        0.66664 .OF. INSTAL ID ASG         ;GET NAME      ;GET BE        ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET
  CAT      ;DO  LAST    00007 RDAL       0.0      TIMES ;DO  LAST    00017 RDAL       7.0       TIMES ;DO  LAST    00022 RDAL     9999
  .0      TIMES

   PATH NUMBER =     12
   ;GET DIRNAME ESG   ;GET NEXT =       1.0       .OF. NAME VAL ESG ;GET NEXT =       1.0       .OF. INAMEVAL LSG ;GET INAMEVAL ASG
   ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET NEXT =       1.0       .OF. ICATVAL ASG   ;GET CAT      ;DO  LAST
   00002 RDAL       0.0      TIMES ;DO  LAST    00008 RDAL       0.0      TIMES ;DO  LAST    00010 RDAL       1.0     TIMES ;GET
  NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPCAT ESG     ;GET NEXT =       1.0       .OF. CATEGORY LSG ;GET CAT ID ASG      ;GET
  CAT2 ESG      ;GET NEXT =       1.0       .OF. CATBK LSG       ;GET NEXT =       0.00002 .OF. INSTAL ID ASG          ;GET NAME      ;GET
  BE      ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT       ;DO  LAST    00007 RDAL       0.0      TIMES ;GET NEXT =      0.66664
```

Table 6. (cont).

```
                           DIAMS PACER DEMONSTRATION                                             REPORT 12 PAGE    3
RDAL TABULATION

  .OF. INSTAL ID ASG      ;GET NAME   ;GET BE     ;GET BESUF  ;GET MAJ     ;GET MIN     ;GET CAT     ;DO  LAST    00007 RDAL
     0.0      TIMES ;DO   LAST    00017 RDAL       39.0      TIMES ;DO  LAST     00021 RDAL     1999.0      TIMES

   PATH NUMBER =     13
   ;GET DIRNAME ESC   ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0     .OF. INAMEVAL LSG ;GET INAMEVAL ASG
   ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET NEXT =      1.0      .OF. ICATVAL ASG  ;GET CAT     ;DO  LAST
   00002 RDAL       0.0      TIMES ;GET  SAME   .OF. INAMEVAL ASG ;GET NAME     ;DO  LAST    00002 RDAL       0.0      TIMES ;DO
  LAST   00011 RDAL       0.0      TIMES ;DO  LAST    00013 RDAL       1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG

   PATH NUMBER =     14
   ;GET DIRNAME ESG  ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0     .OF. INAMEVAL LSG ;GET INAMEVAL ASG
   ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET NEXT =      1.0      .OF. ICATVAL ASG  ;GET CAT      ;DO  LAST
   00002 RDAL       0.0      TIMES ;GET  SAME   .OF. DMASTER3 LSG ;GET INSTAL ID ASG        ;GET NEXT =      1.0     .OF. INAMEV
  AL ASG ;GET NAME     ;DO  LAST    00002 RDAL       0.0      TIMES ;DO  LAST    00013 RDAL      0.0      TIMES ;DO  LAST   00018
   RDAL       1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG

   PATH NUMBER =     15
   ;GET DIRNAME ESG  ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0      .OF. INAMEVAL LSG ;GET NEXT =     1
  .0      .OF. INAMEVAL ASG ;GET NAME     ;DO  LAST    00002 RDAL       0.0      TIMES ;DO  LAST    00004 RDAL       0.0      TIMES
   ;DO  LAST    00006 RDAL       1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPINSTAL ESG         ;GET NEXT =
      1.0      .OF. SUPI LSG       ;GET AI RMKS LSG  ;GET COLL LSG    ;GET ELINS LSG    ;GET CRD LSG      ;GET FUNC LSG
   ;GET NEXT =      0.00002 .OF. INSTAL ID ASG       ;GET NAME    ;GET BE      ;GET BESUF  ;GET MAJ      ;GET MIN      ;GET CAT
   ;DO  LAST    00007 RDAL       0.0      TIMES ;GET NEXT =      0.66664 .OF. INSTAL ID ASG       ;GET NAME    ;GET BE     ;GET
  BESUF  ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL       0.0      TIMES ;DO  LAST    00022 RDAL  79999.0
   TIMES

   PATH NUMBER =     16
   ;GET DIRNAME ESG  ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0      .OF. INAMEVAL LSG ;GET NEXT =     1
  .0      .OF. INAMEVAL ASG ;GET NAME     ;DO  LAST    00002 RDAL       0.0      TIMES ;GET  SAME   .OF. INAMEVAL ASG ;GET DMASTE
  R1 LSG  ;GET DMASTER2 LSG ;GET DMASTER3 LSG  ;GET NEXT =      1.0      .OF. INSTAL ID ASG       ;GET NAME    ;GET BE     ;GET
  BESUF  ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL       0.0      TIMES ;DO  LAST    00016 RDAL      0.0
   TIMES ;DO  LAST    00018 RDAL       1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG

   PATH NUMBER =     17
   ;GET DIRNAME ESG  ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0      .OF. INAMEVAL LSG ;GET NEXT =     1
  .0      .OF. INAMEVAL ASG ;GET NAME     ;DO  LAST    00002 RDAL       0.0      TIMES ;DO  LAST    00004 RDAL       0.0      TIMES
   ;DO  LAST    00006 RDAL       1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPINSTAL ESG        ;GET NEXT =
      1.0      .OF. SUPI LSG       ;GET AI RMKS LSG  ;GET IN ESG ;GET NEXT =      1.0      .OF. IN LSG ;GET INSTAL UNIT ASG
   ;GET NEXT =      0.0      .OF. INSTAL ID ASG       ;GET NAME    ;GET BE      ;GET BESUF  ;GET MAJ      ;GET MIN      ;GET CAT
   ;DO  LAST    00007 RDAL       0.0      TIMES ;GET NEXT =      0.13332 .OF. INSTAL ID ASG       ;GET NAME    ;GET BE ;GET
  BESUF  ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL       0.0      TIMES ;DO  LAST    00018 RDAL      3.99976
   TIMES ;DO  LAST    00022 RDAL  79999.0      TIMES

   PATH NUMBER =     18
   ;GET DIRNAME ESG  ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0      .OF. INAMEVAL LSG ;GET NEXT =     1
  .0      .OF. INAMEVAL ASG ;GET NAME     ;DO  LAST    00002 RDAL       0.0      TIMES ;DO  LAST    00004 RDAL       0.0      TIMES
   ;DO  LAST    00006 RDAL       1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPCPX ESG     ;GET NEXT =     1
  .0      .OF. COMPLEX LSG  ;GET COMPLEX REC ASG       ;GET MAJVAL ESG   ;GET NEXT =      1.0      .OF. COMPBK LSG  ;GET NEXT =
      0.00002 .OF. INSTAL ID ASG       ;GET NAME    ;GET BE      ;GET BESUF  ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST
```

Table 6. (cont).

```
                                        DIAMS PACER DEMONSTRATION                                     REPORT 12 PAGE    4
RDAL TABULATION

 00007 RDAL     0.0      TIMES ;GET NEXT =       0.66664 .OF. INSTAL ID ASG       ;GET NAME   ;GET BE     ;GET BESUF ;GET
MAJ    ;GET MIN     ;GET CAT     ;DO  LAST    00007 RDAL      0.0      TIMES ;DO  LAST   00017 RDAL      9.0     TIMES ;DO
LAST   00021 RDAL   7999.0       TIMES

 PATH NUMBER =      19
 ;GET DIRNAME ESG  ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =       1.0      .OF. INAMEVAL LSG ;GET NEXT = 1
.0      .OF. INAMEVAL ASG ;GET NAME    ;DO  LAST   00002 RDAL      0.0      TIMES ;DO  LAST   00004 RDAL      0.0      TIMES
 ;DO  LAST   00006 RDAL       1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPER AREA ESG     ;GET NEXT =
    1.0      .OF. SUPAREA LSG  ;GET AREATIE LSG  ;GET AREA SLIDE ID ASG  ;GET WACVAL ESG    ;GET NEXT =      1.0      .OF.
WACBA LSG     ;GET NEXT =       0.00002 .OF. INSTAL ID ASG        ;GET NAME     ;GET BE      ;GET BESUF   ;GET MAJ    ;GET MIN
 ;GET CAT     ;DO  LAST    00007 RDAL       0.0      TIMES ;GET NEXT =       0.66664 .OF. INSTAL ID ASG       ;GET NAME    ;GET
BE      ;GET BESUF  ;GET MAJ     ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL      0.0      TIMES ;DO  LAST   00017 RDAL
    7.0      TIMES ;DO  LAST   00022 RDAL    9999.0       TIMES

 PATH NUMBER =      20
 ;GET DIRNAME ESG  ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =       1.0      .OF. INAMEVAL LSG ;GET NEXT =      1
.0      .OF. INAMEVAL ASG ;GET NAME    ;DO  LAST   00002 RDAL      0.0      TIMES ;DO  LAST   00004 RDAL      0.0      TIMES
 ;DO  LAST   00006 RDAL       1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPCAT ESG     ;GET NEXT =      1
.0      .OF. CATEGORY LSG ;GET CAT ID ASG      ;GET CAT2 ESG       ;GET NEXT =       1.0      .OF. CATBK LSG     ;GET NEXT =  0
.00002 .OF. INSTAL ID ASG       ;GET NAME      ;GET BE      ;GET BESUF ;GET MAJ     ;GET MIN   ;GET CAT     ;DO  LAST    00007
RDAL      0.0      TIMES ;GET NEXT =       0.66664 .OF. INSTAL ID ASG       ;GET NAME    ;GET BE     ;GET BESUF  ;GET MAJ
 ;GET MIN     ;GET CAT     ;DO  LAST    00007 RDAL       0.0      TIMES ;DO  LAST   00017 RDAL     39.0      TIMES ;DO  LAST
00021 RDAL   1999.0       TIMES

 PATH NUMBER =      21
 ;GET DIRNAME ESG  ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =       1.0      .OF. INAMEVAL LSG ;GET INAMEVAL ASG
 ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET INSTAL ID ASG        ;GET NEXT =       1.0      .OF. INAMEVAL ASG
 ;GET NAME    ;DO  LAST    00002 RDAL       0.0      TIMES ;DO  LAST    00009 RDAL      0.0      TIMES ;DO  LAST   00011 RDAL
     1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPINSTAL ESG        ;GET NEXT =       1.0      .OF. SUPI L
SG      ;GET AI RMKS LSG  ;GET COLL LSG     ;GET ELINS LSG    ;GET CRD LSG         ;GET FUNC LSG      ;GET NEXT =     0.00002
 .OF. INSTAL ID ASG       ;GET NAME    ;GET BE      ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST   00007 RDAL
    0.0      TIMES ;GET NEXT =       0.66664 .OF. INSTAL ID ASG       ;GET NAME    ;GET BE      ;GET BESUF   ;GET MAJ      ;GET
MIN      ;GET CAT      ;DO  LAST    00007 RDAL       0.0      TIMES ;DO  LAST    00022 RDAL   79999.0       TIMES

 PATH NUMBER =      22
 ;GET DIRNAME ESG  ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =       1.0      .OF. INAMEVAL LSG ;GET INAMEVAL ASG
 ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET INSTAL ID ASG        ;GET NEXT =       1.0      .OF. INAMEVAL ASG
 ;GET NAME    ;DO  LAST    00002 RDAL       0.0      TIMES ;GET  SAME  .OF. INSTAL ID ASG        ;GET NAME     ;GET BE      ;GET
BESUF  ;GET MAJ     ;GET MIN      ;GET CAT      ;DO  LAST    00007 RDAL       0.0      TIMES ;DO  LAST    00017 RDAL       0.0
 TIMES ;DO  LAST    00019 RDAL        1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG

 PATH NUMBER =      23
 ;GET DIRNAME ESG  ;GET NEXT =       1.0      .OF. NAME VAL ESG ;GET NEXT =       1.0      .OF. INAMEVAL LSG ;GET INAMEVAL ASG
 ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET INSTAL ID ASG        ;GET NEXT =       1.0      .OF. INAMEVAL ASG
 ;GET NAME    ;DO  LAST    00002 RDAL       0.0      TIMES ;DO  LAST    00009 RDAL      0.0      TIMES ;DO  LAST   00011 RDAL
     1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPINSTAL ESG        ;GET NEXT =       1.0      .OF. SUPI L
SG      ;GET AI RMKS LSG  ;GET IN ESG ;GET NEXT =       1.0      .OF. IN LSG ;GET INSTAL UNIT ASG     ;GET NEXT =      0.0
 .OF. INSTAL ID ASG       ;GET NAME    ;GET BE      ;GET BESUF   ;GET MAJ      ;GET MIN      ;GET CAT      ;DO  LAST   00007 RDAL
    0.0      TIMES ;GET NEXT =       0.13332 .OF. INSTAL ID ASG       ;GET NAME    ;GET BE      ;GET BESUF   ;GET MAJ      ;GET
MIN      ;GET CAT      ;DO  LAST    00007 RDAL       0.0      TIMES ;DO  LAST    00018 RDAL      3.99976 TIMES ;DO  LAST   00022
```

Table 6. (concl).

```
                                        DIAMS PACER DEMONSTRATION                                   REPORT 12 PAGE
RDAL TABULATION

  RDAL  79999.0     TIMES

  PATH NUMBER =    24
  ;GET DIRNAME ESG  ;GET NEXT =      1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0     .OF. INAMEVAL LSG ;GET INAMEVAL ASG
  ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET INSTAL ID ASG       ;GET NEXT =      1.0      .OF. INAMEVAL ASG
  ;GET NAME     ;DO  LAST    00002 RDAL        0.0       TIMES ;DO   LAST    00009 RDAL        0.0       TIMES ;DO   LAST    00011 RDAL
      1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPCPX ESG    ;GET NEXT =      1.0      .OF. COMPLEX LSG
  ;GET COMPLEX REC ASG     ;GET MAJVAL ESG    ;GET NEXT =      1.0      .OF. COMPBK LSG    ;GET NEXT =      0.00002 .OF. INSTAL
  ID ASG       ;GET NAME     ;GET BE     ;GET BESUF  ;GET MAJ      ;GET MIN      ;GET CAT     ;DO   LAST    00007 RDAL        0.0
  TIMES ;GET NEXT =      0.66664 .OF. INSTAL ID ASG         ;GET NAME     ;GET BE      ;GET BESUF  ;GET MAJ      ;GET MIN      ;GET
 CAT    ;DO  LAST    00007 RDAL        0.0       TIMES ;DO   LAST    00017 RDAL        9.0       TIMES ;DO   LAST    00021 RDAL     7999
 .0     TIMES

  PATH NUMBER =    25
  ;GET DIRNAME ESG  ;GET NEXT =      1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0     .OF. INAMEVAL LSG ;GET INAMEVAL ASG
  ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET INSTAL ID ASG       ;GET NEXT =      1.0      .OF. INAMEVAL ASG
  ;GET NAME     ;DO  LAST    00002 RDAL        0.0       TIMES ;DO   LAST    00009 RDAL        0.0       TIMES ;DO   LAST    00011 RDAL
      1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPER AREA ESG      ;GET NEXT =      1.0      .OF. SUPARE
 A LSG   ;GET AREATIE LSG  ;GET AREA SLIDE ID ASG  ;GET WACVAL ESG    ;GET NEXT =      1.0      .OF. WACBK LSG     ;GET NEXT =
      0.00002 .OF. INSTAL ID ASG          ;GET NAME     ;GET BE      ;GET BESUF  ;GET MAJ      ;GET MIN      ;GET CAT     ;DO  LAST
  00007 RDAL        0.0       TIMES ;GET NEXT =      0.66664 .OF. INSTAL ID ASG         ;GET NAME     ;GET BE      ;GET BESUF  ;GET
 MAJ     ;GET MIN      ;GET CAT      ;DO   LAST    00007 RDAL        0.0       TIMES ;DO   LAST    00017 RDAL        7.0       TIMES ;DO
 LAST    00022 RDAL    9999.0      TIMES

  PATH NUMBER =    26
  ;GET DIRNAME ESG  ;GET NEXT =      1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0     .OF. INAMEVAL LSG ;GET INAMEVAL ASG
  ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET INSTAL ID ASG       ;GET NEXT =      1.0      .OF. INAMEVAL ASG
  ;GET NAME     ;DO  LAST    00002 RDAL        0.0       TIMES ;DO   LAST    00009 RDAL        0.0       TIMES ;DO   LAST    00011 RDAL
      1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG ;GET SUPCAT ESG    ;GET NEXT =      1.0      .OF. CATEGORY LSG
  ;GET CAT ID ASG    ;GET CAT2 ESG      ;GET NEXT =      1.0      .OF. CATBK LSG     ;GET NEXT =      0.00002 .OF. INSTAL ID AS
 G       ;GET NAME     ;GET BE      ;GET BESUF  ;GET MAJ      ;GET MIN      ;GET CAT     ;DO   LAST    00007 RDAL        0.0       TIMES
  ;GET NEXT =      0.66664 .OF. INSTAL ID ASG         ;GET NAME     ;GET BE      ;GET BESUF  ;GET MAJ      ;GET MIN      ;GET CAT
  ;DO   LAST    00007 RDAL        0.0       TIMES ;DO   LAST    00017 RDAL       39.0       TIMES ;DO   LAST    00021 RDAL    1999.0
  TIMES

  PATH NUMBER =    27
  ;GET DIRNAME ESG  ;GET NEXT =      1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0     .OF. INAMEVAL LSG ;GET NEXT =       1
 .0        .OF. INAMEVAL ASG ;GET NAME     ;DO   LAST    00002 RDAL        0.0       TIMES ;GET   SAME  .OF. INAMEVAL ASG ;GET DMASTE
 R1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET NEXT =      1.0      .OF. ICATVAL ASG   ;GET CAT      ;DO   LAST    00002 RDAL
      0.0      TIMES ;DO   LAST    00011 RDAL        0.0       TIMES ;DO   LAST    00013 RDAL        1.0       TIMES ;GET NEXT = 53331
 .82812 .OF. NAME VAL ESG

  PATH NUMBER =    28
  ;GET DIRNAME ESG  ;GET NEXT =      1.0      .OF. NAME VAL ESG ;GET NEXT =      1.0     .OF. INAMEVAL LSG ;GET INAMEVAL ASG
  ;GET DMASTER1 LSG ;GET DMASTER2 LSG ;GET DMASTER3 LSG ;GET INSTAL ID ASG       ;GET NEXT =      1.0      .OF. INAMEVAL ASG
  ;GET NAME     ;DO  LAST    00002 RDAL        0.0       TIMES ;GET   SAME  .OF. DMASTER3 LSG ;GET NEXT =      1.0      .OF. ICATVA
 L ASG   ;GET CAT      ;DO   LAST    00002 RDAL        0.0       TIMES ;DO   LAST    00013 RDAL        0.0       TIMES ;DO   LAST    00015
  RDAL       1.0      TIMES ;GET NEXT = 53331.82812 .OF. NAME VAL ESG
```

CONCLUSIONS

Applicability of Relational Model

We found that the relational model forms a very satisfactory basis for modeling the information content of an existing database. We also found that considerable analysis of existing record types and associations is necessary. An even greater analysis of the problem's semantics and operations of applications programs would be required to discover and normalize functional dependencies. The approach of considering record types as basic candidates for relational definitions, then performing various modifications, worked well. We encountered no difficulty in representing one-to-one records, many-to-one records, or many-to-many records using the methods summarized under Relational Model.

Applicability of Access Path Model

We found that the string level of DIAM I as modified by Schneider[2] forms a satisfactory basis for modeling most of the implementations of access paths. It was found suitable for:

1) Simple and compound collections of attributes;
2) Master/detail chaining, including the linked-master feature;
3) The major logical features of working directories and master/argument directories;
4) Subsetting and ordering of instances;
5) Entry points--either direct or hashed.

Certain implementation features proved difficult to model with fidelity within the string level. Such features include:

1) Multiple detail record types attached to a single chain--we must consider these as separate links;
2) Tertiary indexing based on limit-page directories and similar techniques that use information about the storage structure. We consider these as features to consider at the encoding level of DIAM;
3) Cases in which some instances of a relation are accessed by one type of structure and other instances by another type of structure--or no structure at all. The extension of the original theory proposed by Schneider[2] using the "restriction-distribution" concept is applicable to these cases. However, the current version of the RIPS software does not have this concept completely implemented.

---

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

One source of complication in moving from an existing implementation to a model is trying to ascertain the dividing line between the access path model and encoding model. The analyst has some flexibility in this regard, as pointed out under Modeling of Directories in discussing the alternatives for the Argument/Master Directory system.

Applicability of Existing Components of RIPS Software

Overall Evaluation - The size of this application was an order of magnitude larger than any previously executed on the software. We were gratified to observe that the large amount of input data was properly accepted and operated on. This was also the first application executed under the IBM Model 370 version of the software; previous executions operated on CDC and Univac mainframes. The IBM version permitted both virtual storage and the use of certain space economies resulting from the machine's byte orientation and the FORTRAN compiler.

With few exceptions (noted below), the software performed just as specified and produced the sets of results desired.

Minor Corrections Needed - This demonstration exercised portions of the software for the first time and disclosed errors and occasions of design that we feel should be improved or corrected. These include:

1) Lack of allowable precision in the numeric portion of the Matching Criterion parameter of the L-string. This value should be converted from integer to floating-point representation. We note that this parameter applies only in the simulation mode of the software for the purpose of calculating qualified instances;

2) Error in the logic that computes the number of qualified instances when the path leaves the environment of one relation, moves to another, and returns to the first. The qualification parameter now is applied twice to result in erroneous values for the number of instances qualified;

3) Error in the logic to generate correct RDAL in some cases in which more than one "same string" entry appears in a single path;

4) Error in instances traversed when the number of instances qualified is less than one.

Correction of these errors entails only a small amount of analysis followed by care in reprogramming a few routines. There are no underlying theoretical problems.

Desirable Extensions -

1) Extension of the uniform distribution situation to more complex distributions such as "normal," "Zipfian," or "empirical;"

2) More complete statistical methods in the determination of instances qualified, traversed, and present;

3) Completion of logic to handle the previously defined "restriction-distribution";

4) Better use of partially defined queries through addition of DSN-to-DSN rates and distributions;

5) Extension of function capabilities to allow for designation of both functions (composed of RNs or other functions) and RNs in queries.

Requirements on Information Gathering

This study has highlighted the fact that initial data gathering and understanding of the application are important and difficult parts of the problem. In any database study--whether for a system design or for use as a relational interface--the complete statement of underlying entities, their semantics, and the projected workload is essential. Because of the classified nature of much of this information in our case, we were obliged to make assumptions and simplifications.

The methodology and discipline described by Schneider and Spath as "Quantitative Data Description"[12] can provide a formal detailed approach to collecting and interpreting the defining data.

12. 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).

REFERENCES

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

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

3. Planning Research Corporation: *Program Assisted Console Evaluation and Review (PACER) System Description*. RADC-TR-72-89, Vol I, Apr 72 (910395L)

4. W. W. Gaertner Research Inc.: *Final Report on Analysis of Requirements for Large-Scale Multi-User Intelligence Data-Base Processing System (DBPS)*. Vol II, WWGRI-4056-770210, February 10, 1977.

5. PRC Information Sciences Co.: *PACER Programming Language Specifications*. TM 003, September 1975.

6. PRC Information Sciences Co.: Unpublished notes and tutorial material, January 1978.

7. 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.

8. 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).

9. 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.).

10. 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.

11. 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.

12. 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).

APPENDIX A

A Note on Evolving Terminology

Terminology in the field of database modeling is still evolving. In this report, we have therefore used a mixture of terms that are not wholly consistent. To assist the reader in interpreting and equating various terms, we present the following summary.

The DIAM model[1] established four levels of abstractions for analysis of database systems, of which we are interested in the upper two--the information level and the access path level. For the information level in DIAM, Senko adopted the Entity Set Model. He introduced and defined the "string" model as the basis for the access path level.

The Relational Model[7] deals exclusively with the information level. It uses the underlying terms from relational mathematics.

The relational-DIAM approach[8] recognized that the inherent compatibility of the two basic model and proposed a consolidation. The consolidation would graft the relational preciseness of Codd onto the remaining levels of the DIAM structure and reconcile the terminology.

The various terms from these models are shown in Table A-1. In this report, we have used the following.

Information Level - In the narrative sections of this report, we have adopted the relational terminology, mainly because of its broader use in the literature. However, our computer simulation model was programmed using the Entity Set terminology of the original DIAM. Therefore, input and output formats are in that notation. The reader will therefore find in the printouts references to DSNs, ESNs, and RNs; an exception is Report 23, which was programmed more recently specifically to give a relational representation.

Access Path Level - In this report, the original DIAM string notation appears in computer input and output formats and in diagrams. The Relational DIAM notation appears in the narrative.

---

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

7. 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.

8. 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).

Table A-1. Equivalent terminology among various models.

	DIAM (Senko)[1]	Relational (Codd)[7]	Relational DIAM (Schneider)[2]
Information Level	Description Set Name (DSN)	Relation	Relation
	Entity Set Name (ESN)	Domain	Domain
	Role Name (RN)	Attribute	Attribute
Access Path Level	Atomic String (ASG)	(none defined)	Projection String (PSG)
	Entity String (ESG)		Restriction String (RSG)
	Link String (LSG)		Join String (JSG)

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

7. 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.

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

GLOSSARY OF TERMS

CPU	- computer processing unit
DBMS	- database management system
DIAM	- data-independent accessing model
DML	- data manipulation language
DSN	- description set name
DSS	- directory services system
ESN	- entity set name
JSG	- join string
MC	- matching criterion (criteria)
OO	- order on
PACER	- program-assisted console evaluation and review
PSG	- projection string
RDAL	- representation-dependent accessing language
RDMS	- reentrant data managment system
RIAL	- representation-independent accessing language
RIPS	- representation-independent programming system
RN	- role name
RSG	- restriction string
SSC	- subset selection criterion (criteria)