



3 4456 0056279 0

ORNL/TM-9696

# oml

OAK RIDGE  
NATIONAL  
LABORATORY

MARTIN MARIETTA

## Evaluation of Relational Database Products for the VAX

K. L. Kannan

OAK RIDGE NATIONAL LABORATORY

CENTRAL RESEARCH LIBRARY

CIRCULATION SECTION

4309N, P.O. BOX 117

**LIBRARY LOAN COPY**

DO NOT TRANSFER TO ANOTHER PERSON

If you wish someone else to see this  
report, send in name with report and  
the library will arrange a loan.

OPERATED BY  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
FOR THE UNITED STATES  
DEPARTMENT OF ENERGY



Fusion Energy Division

**EVALUATION OF RELATIONAL DATABASE  
PRODUCTS FOR THE VAX**

**K. L. Kannan**

Computing and Telecommunications Division

Date Published - November 1985

**NOTICE:** This document contains information of a preliminary nature. It is subject to revision or correction and therefore does not represent a final report.

Prepared by  
OAK RIDGE NATIONAL LABORATORY  
Oak Ridge, Tennessee 37831  
operated by  
MARTIN MARIETTA ENERGY SYSTEMS, INC.  
for the  
U.S. DEPARTMENT OF ENERGY  
under Contract No. DE-AC05-84OR21400



3 4456 0056279 0



The following are trademarks of Digital Equipment Corporation:

ACMS

CDD

DATATRIEVE

DEC

DECnet

PDP

Rdb/ELN

Rdb/VMS

TDMS

VAX

VAX Information Architecture

VMS

The following are trademarks of Software House:

System 1032

System 1022

The following are trademarks of Relational Technology Inc.:

INGRES

EQUEL

QUEL



# CONTENTS

ACKNOWLEDGMENTS . . . . .	vii
ABSTRACT . . . . .	ix
1. INTRODUCTION . . . . .	1
2. PRODUCT DESCRIPTION . . . . .	3
3. METHOD . . . . .	5
4. OPERATING ENVIRONMENT . . . . .	7
5. PHASE I . . . . .	8
5.1. DATABASE DESIGN . . . . .	8
5.2. DATABASE DEFINITION AND STORAGE . . . . .	10
5.2.1. INGRES . . . . .	10
5.2.2. Rdb . . . . .	11
5.2.3. S1032 . . . . .	15
5.2.4. Performance and Size . . . . .	15
5.3. OPERATIONS . . . . .	17
6. PHASE II . . . . .	43
6.1. DATABASE DESIGN . . . . .	43
6.2. DATABASE DEFINITION AND STORAGE . . . . .	44
6.3. OPERATIONS . . . . .	46
6.3.1. Selection and Projection . . . . .	46
6.3.2. Joins . . . . .	53
6.3.3. Maintenance Operations . . . . .	55
7. GENERAL OBSERVATIONS AND COMPARISON OF FEATURES . . . . .	59



## ACKNOWLEDGMENTS

With the increasing number of VAX processors and the proliferation of available products for these machines, there is a great need for software evaluation. An active area of product development mirroring consumer interest and need is database management. Database management systems based on the relational data model are of particular interest because of their ease of use and simplicity in database design. Even in this relatively new field there are a number of alternative products.

The Fusion Energy Division of Oak Ridge National Laboratory (ORNL) encouraged and supported this evaluation of relational database products. It very generously made the necessary machine resources available as well as supported the investigation and the writing of this report.

Software was provided at little or no cost by the vendors: Digital Equipment Corporation (Rdb), Relational Technology Inc. (INGRES), and Software House (S1032). They were responsive and supportive throughout this study.

Phase I of the evaluation was a cooperative effort with Princeton Plasma Physics Laboratory (PPPL). The author expresses appreciation to Dr. Stan Kaye of PPPL for the design of a magnetic fusion energy database and operations pertinent to such data.

In the second phase of the evaluation, participation from Martin Marietta Energy Systems, Inc. VAX users and personnel with interest in database management was invited. Jeri McNeany and Jack Jones of the Computing and Telecommunications Division and Ann Stewart of the Fusion Energy Division contributed to the design of the Phase II test database and the development of a battery of operations to use in the evaluation. I gratefully acknowledge the further participation of Jeri McNeany who translated the test operations into the proper syntax for both S1032 and S1022.

This project was initiated by Dr. W. R. Wing of the Fusion Energy Division. His suggestions, ideas, and discussions were important contributions to this work. The author thanks him.



## ABSTRACT

Four commercially available database products for the VAX/VMS operating system were evaluated for relative performance and ease of use. The products were DATATRIEVE, INGRES, Rdb, and S1032. Performance was measured in terms of elapsed time, CPU time, direct I/O counts, buffered I/O counts, and page faults. Ease of use is more subjective and has not been quantified here; however, discussion and tables of features as well as query syntax are included. This report describes the environment in which these products were evaluated and the characteristics of the databases used. All comparisons must be interpreted in the context of this setting.



## 1. INTRODUCTION

This study was undertaken because of the increasing investment in VAX computers by Martin Marietta Energy Systems, Inc., and the use of these computers for database activities. As in many other product areas, there is a need for software evaluation. Even in the relatively new field of relational database management, limited further for the VAX/VMS operating system, there are a number of alternative products.

This report describes the evaluation of four commercially available relational database products for the VAX/VMS operating system. The study was not an exhaustive test of the products, but it did exercise the relational operators SELECT, PROJECT, and JOIN and the relational capability of dynamic restructuring. The work was not intended to be an attempt to rank the products. Results from an evaluation such as this are very dependent upon the version of the software used, database design, technique employed, and operating environment. Accordingly, all of these factors will be described in detail so that the results reported here may be useful in predicting how each of the products will perform with the reader's data in the reader's environment.

The products included in this investigation are DATATRIEVE, INGRES, Rdb/VMS(Rdb), and System 1032 (S1032). They were evaluated for relative performance and ease of use. Performance was measured in terms of elapsed time, CPU time, direct I/O counts, buffered I/O counts, and page fault counts.

Ease of use is more subjective. Among the things to consider are documentation, interactive assistance, data manipulation language, programming language interface, technical support, and special features such as interactive forms and graphics.

More important than the ease-of-use issue is ease of use with efficiency. Each product included in this investigation was judged easy to use. By using the vendor-provided learning tools, a person without prior training in the product could quickly and easily make use of the product. Judgments of efficiency are based on tabulated performance figures and discussion of features and capabilities.

There were two phases to this study. Phase I was a cooperative effort with other Department of Energy magnetic fusion energy laboratories in an attempt to determine what database management system(s) would be most compatible with and useful in an experimental physics environment. Particular emphasis was placed on numerical data, host language interface, and performance. INGRES, Rdb, and S1032 were included in this phase.

Phase II reflected the concern within Energy Systems with migration from the PDP-10 to the VAX. Of primary interest was the accommodation of data and operations of the kind handled by System 1022 (S1022). Datasets from S1022 were used in the design of the Phase II database. Operations were executed on the database in S1022 as well as in the VAX systems. In addition

## 2 *Introduction*

to INGRES, Rdb, and S1032, DATATRIEVE—a product currently used on many Energy Systems' VAXs—was included.

Section 2 of this report gives a brief description of the four database products. The technique employed in evaluating relative performance is presented in Sect. 3. Section 4 contains a description of the operating environment. Phase I database design, operations, and results are presented in Sect. 5, and those of Phase II, in Sect. 6. General observations and a comparison of features are discussed in the final section.

There is currently much interest in the technique of benchmarking, with studies under way and papers appearing in the literature. I have refrained from using the term benchmark, which presumes a standard; I know of no such measurement for relational database performance. Moreover, a better indicator of the usefulness of database management systems is their performance of realistic functions on an actual database; that is the basis of this study.

## 2. PRODUCT DESCRIPTION

Although the primary interest was in relational database management systems, this study included S1032, which is relational-like rather than relational, and DATATRIEVE, which is not a database management system but a query language and report generator with relational capabilities. The other two products, INGRES and Rdb, are fully functional relational database management systems.

It is important to note the version of each product since evaluation results are dependent upon the software, are valid for one point in time, and can be expected to change with product development. Results reported herein pertain to the following:

DATATRIEVE	v3.0
INGRES	v2.1
Rdb	v1.0
S1032	v3.0

DATATRIEVE, from DEC, is a component of the VAX Information Architecture with interfaces to the other components. As stated above, it is not a database management system. It has no unique access method but uses that of the interfacing system: Record Management Services (RMS), Rdb, or DBMS, a CODASYL(Conference on Data Systems Languages)-compliant database management system by DEC. In Phase II of this evaluation, DATATRIEVE was used as the query language with RMS indexed sequential access method (ISAM) files and with Rdb. In the following sections, DATATRIEVE, when it appears alone, indicates use with RMS. Rdb is always specified when discussion applies to it.

INGRES, from Relational Technology Inc., was developed as a research prototype at the University of California, Berkeley, in the early to mid-70s under UNIX. It has been available for VMS since 1981. The INGRES system is written in C. It provides a layered architecture with the base layer being the relational database management system and its language, QUEL. Upper layers consist of several kinds of user interfaces: query by forms, report by forms, graph by forms, and application by forms. Four storage structures are available: Heap (sequential), ISAM, Hash, and with v3.0 (not included in this study) B-tree, as well as the compressed form of each.

INGRES runs two processes that communicate entirely through mailboxes. When INGRES is invoked, a database is indicated and a backend process is spawned which couples with the database and handles all the database access. INGRES accommodates multiple databases, but only one can be accessed per INGRES process. There are no multiple database operations. This is not necessarily a shortcoming because a database may contain several thousand relations. It does, however, require

#### 4 *Product Description*

more centralized control or coordination: a database must contain all information to be accessed in common.

Rdb from DEC was announced in the spring of 1984 and was shipped in late summer/early fall of 1984. It has full relational functionality, but to achieve the user interface capabilities of the other systems, it is necessary to use the VAX Information Architecture (VIA) products: Common Data Dictionary (CDD), DATATRIEVE, TDMS for forms, and ACMS for application development. Rdb is a component of VIA but can be used in a stand-alone mode without any of the other VIA products. It is optimized for use with a host language rather than its own terminal interface operator RDO or with DATATRIEVE. Two file structures are available: sequential and B-tree. Its implementation of B-tree indexing imposes no order on the data and includes no concept of a primary or clustering index.

In addition to being a member of the VIA product set, Rdb is also a component of DEC's Standard Relational Interface (DSRI), which implies compatibility with a second family of products. Currently, the only other product is Rdb/ELN, a relational database management system for the ELN operating environment used primarily in real-time processing. There are projections and conjectures of future DSRI products such as database servers and database machines. DEC's commitment to a standard relational interface ensures Rdb's compatibility with any such products.

S1032 has been available since early 1983 from Software House, the vendor of S1022. It is not a VMS version of S1022 but is a unique product developed for VAX/VMS, written in macro and making use of VMS features such as asynchronous I/O. S1032 does not adhere strictly to the relational formula: arrays are allowed, all relational operators are not implemented, and its command language is dependent on database structure. Using its procedural language, it is possible to affect the missing join and project (implying uniqueness) operations. There are two file structures in S1032: sequential and B-tree. As with Rdb, all B-tree indexes are created equal.

The above is a brief description of the products. More descriptive information appears in Sects. 5, 6, and 7.

### 3. METHOD

The approach of this study was to base evaluation on expected use. Real data and database operations pertinent to the testing environment were used.

Relational database systems claim ease of use and good performance with minimal tuning. Consequently, vendor-provided documentation served as the learning tool, and default system parameter settings were used unless documentation specified otherwise. This approach may not yield optimum performance but that was not the goal of this study. The aim was to determine relative performance for expected use.

The implementation plan for the evaluation was for a group effort in designing a database and defining a battery of operations followed by each participant working independently with all systems. Unfortunately, time constraints, priorities, and work assignments permitted only limited participation by other members of the group.

The set of operations began with defining and loading the database and continued through the exercise of all relational operations. Database design incorporated all common VMS datatypes, records of different sizes, and relations with different numbers of records. Operations were run numerous times under controlled conditions, with the average of the three best trials being the result recorded. Performance measurements were obtained using VAX Run-Time Library (RTL) procedures accessed through the RTL interface provided by the database systems. Results are those obtained in a single-user environment. A Digital Equipment Corporation (DEC) product, VAX-11 SPM, System Performance Monitor, sampled the system and recorded system resource utilization during all tests. SPM reports were used in the interpretation of performance figures. Operations were also repeatedly run in a multiuser environment with the machine devoted to multiple processes all doing database operations in one database system with SPM monitoring the system.

The performance in a multiuser environment varied greatly depending on the type of operation, whether CPU- or I/O-intensive. The measurement affected was the elapsed time; all other performance measurements remained essentially the same as in the single-user mode. For CPU-intensive operations and five users of different databases or different operations with the same database, response time was seen to increase by approximately 300%. This was true for all four systems evaluated. For I/O-intensive operations, the increase in response time, although less, was still quite significant: 100-200% for five users of different databases or the same database but different operations. When all processes were executing the same type of operation, each process got approximately the same proportion of the CPU. The relative positions in performance ranking established in the single process tests were maintained in the multiuser tests.

To determine the impact of these database products on other processes, a CPU-intensive job of known activity was run while database operations were being executed. Again, the result was

## 6 *Method*

dependent on the type of database operation. Similar results were found for all four systems evaluated. Against one database process, CPU usage by the competing process for a fixed time was reduced by 50% for high I/O operations. With five database processes executing, the CPU usage for the competing process was reduced by 85–88%.

## 4. OPERATING ENVIRONMENT

The operating environment is an important consideration in performance. One cannot expect the same performance when the products are tested on a VAX 11/730 with two megabytes of memory and a VAX 11/780 with eight megabytes or when the working set size is 150 pages versus one of 1024 pages.

Figure 4.1 shows the operating environment of this study.

.....

VAX 11/780 with seven megabytes of memory  
VMS v3.7  
RP07 disk drive

    User quotas

PRCLM	3	WSQUOTA	400*
ASTLM	20	WSEXTENT	1024*
ENQLM	600*	BIOLM	6
TQELM	10	DIOLM	6
BYTLM	24486	FILLM	50*
PBYTLM	0	SHRFILLM	0
WSDEFAULT	400*	PGFLQUOTA	10000

\* Increased from system default settings

**Fig. 4.1. Operating environment for testing database products.**

.....

The user quotas are not necessarily the optimal settings. They are the ones recommended in products' documentation. In the case of working set size, it may be advisable to set WSQUOTA slightly higher (512). Because of the controlled conditions of this study, the database process was always able to get a working set size of 1024 pages (WSEXTENT).

The four systems were simple to install using the VMSINSTAL procedure. Documentation adequately specified global section, global page, and other VMS system parameter requirements. With each database system, there were a number of special parameters; these were given the default settings recommended in the documentation.

## 5. PHASE I

Prior to this study, a database committee with members from the Department of Energy magnetic fusion energy laboratories had decided to integrate relational database technology into their data acquisition and analysis, functions being done more and more on VAXs. For fairly obvious reasons, such as similarity of environment, sharing of codes, personnel, and data, it was hoped that a common system would be used at the laboratories. With the special constraints and general requirements of this user group in mind, an evaluation of three systems, INGRES, Rdb, and S1032, was undertaken. These systems were selected because of their functionality, availability for evaluation, and compatibility with user experience and existing software.

### 5.1. DATABASE DESIGN

For this phase of the evaluation, a Magnetic Fusion Energy (MFE) database was designed and a set of 16 representative queries defined. The database design is seen in Fig. 5.1.

To determine the impact of database size on performance, the test operations were run on the MFE database at three different sizes representing common database usage. Figure 5.2 shows the number of records per relation.

Data, both real and generated, were loaded from VAX RMS files. The procedure and syntax for defining and loading the MAGSET relation in the three database management systems are discussed in the sections that follow.

---

MFE Database

Relation 1:	26	fields
SUMSET	3	character (Bytes: 4, 8, 100)
	9	integer
	14	floating point
	3	indexes
	2	integer
	1	floating point
Relation 2:	21	fields
NBISSET	2	character (Bytes: 4, 4)
	10	integer
	9	floating point
	2	indexes (integer)
Relation 3:	9	fields
MAGSET	1	integer
	8	floating point
	1	index (integer)

---

**Fig. 5.1. Database design for MFE.**

---

.....

	Number of Records			
	SUMSET	NBISET	MAGSET	TOTAL
1	827	676	827	2330
2	8270	6760	8270	23300
3	33080	27040	33080	93200

**Fig. 5.2. MFE database size.**

.....

## 5.2. DATABASE DEFINITION AND STORAGE

### 5.2.1. INGRES

Before using INGRES, a VAX user must be made an authorized INGRES user by the INGRES system manager or database administrator using the INGRES utility ACCESSDB. This illustrates the point made earlier of central control in INGRES. It is also necessary for a special INGRES account to be created on all devices on which INGRES databases will reside. Databases are stored in subdirectories in the INGRES.DATA directory, not in users' areas.

Prior to invoking INGRES, a user must run the INGRES utility procedure CREATEDB. This is executed at VMS monitor level. The name of a new database is the only input. CREATEDB creates a directory in INGRES.DATA containing 19 files that are INGRES relations and will contain database metadata comprising the data dictionary for INGRES. Figure 5.3 is a directory of the area resulting from "CREATEDB MFE." Data are later stored in this same area, one file per relation.

Figure 5.4 is a listing of the command file used in defining and loading the MAGSET relation. The "copy" command enables INGRES to interface with VAX RMS. With "copy," it is possible to move data **from** an RMS file into an INGRES relation and also to move data **to** an RMS file from an INGRES relation. To copy data from an RMS file, the file must be one of the following types: (1) variable length with carriage return, or (2) fixed length with no carriage control. If not of a correct type, the RMS CONVERT utility can be used to convert the file.

INGRES documentation recommends that data be loaded into a relation with sequential or heap structure. The relation may subsequently be modified to alternative structures with the "modify" command. Figure 5.4 shows that MAGSET's structure was modified from heap to ISAM, resulting in ordering of MAGSET on the integer field SHOTNUM, the primary index.

---

Directory SYS\$SYSDEVICE:[INGRES.DATA.MFE]

ABFAPPL.KK;1	12	9-APR-1985	14:17
ABFOBJS.KK;1	12	9-APR-1985	14:17
ADMIN21;1	8	9-APR-1985	14:17
ATTRIBUTE.KK;1	52	9-APR-1985	14:17
FDFIELDS.KK;1	4	9-APR-1985	14:17
FDFRAMES.KK;1	28	9-APR-1985	14:17
FDTRIMKK;1	4	9-APR-1985	14:17
GCOMMANDS.KK;1	4	9-APR-1985	14:17
GRAPHS.KK;1	28	9-APR-1985	14:17
INDEXES.KK;1	20	9-APR-1985	14:17
INTEGRITL.KK;1	28	9-APR-1985	14:17
PROTECT.KK;1	28	9-APR-1985	14:17
QBFMAP.KK;1	12	9-APR-1985	14:17
RCOMMANDS.KK;1	4	9-APR-1985	14:17
RELATION.KK;1	28	9-APR-1985	14:17
REPORTS.KK;1	12	9-APR-1985	14:17
TREE.KK;1	28	9-APR-1985	14:17
ZOPT1STAT.KK;1	20	9-APR-1985	14:17
ZOPT2STAT.KK;1	20	9-APR-1985	14:17

Total of 19 files, 352 blocks.

**Fig. 5.3. CREATEDB MFE directory.**

---

### 5.2.2. Rdb

In Rdb, all database definition must be done through Rdb's interactive utility procedure, Relational Database Operator (RDO). A command file of RDO commands for defining MAGSET is seen in Fig. 5.5. Since default settings were taken, the three tuning parameters which may be specified in database definition are not shown. The parameters—number of buffers [20], buffer size [3 pages], and page size [2 blocks]—determine the physical storage of the database file. The user is advised to take the default values except in cases such as a record too large for a page or usage dominated by a particular type of transaction. In the latter case, the buffer parameters can be modified for

```

.....

$ createdb mfe
$ ingres mfe

\*      Defining Relation 3, MAGSET, in database MFE      *\

create magset (shotnum=f4, lambda=f4, lihaf=f4,
               betpdi=f4, betpeq=f4, bettdi=f4,
               getteq=f4, tauedi=f4, tauseq=f4)
\g

\*      Storing data in Relation 3 from an RMS file      *\

copy magset (shotnum=c13, lambda=c10, lihaf=c10,
             betpdi=c10, betpeq=c10, bettdi=c10,
             betteq=c10, tauedi=c10, tauseq=c10) from
"sys$userc:[kannan.data]magset3.dat"
\g

modify magset to isam on shotnum
\g

\q

$exit

```

Fig. 5.4. Command file to define and load MAGSET in INGRES.

performance tuning. Some varying of parameter settings was done during this evaluation, and an effect on performance was clearly seen. Increasing the number of buffers and decreasing the buffer size improved random selection, while increasing buffer size and decreasing the number of buffers gave improved performance for projection.

In Rdb, as in INGRES, it is generally preferable to store data into a sequential file and later impose an alternative structure. This was done for all relations except MAGSET, where only one index was used. By doing the loading and indexing both ways (loading with indexing and loading

```

.....

$rdc
define database "sys$sysdvice:[rdb.tokamak]mfe"
    in cdd$top.mfe.
define relation magset.
    shotnum datatype is signed longword.
    lambda datatype is ffloating.
    lihaf datatype is ffloating.
    betpdi datatype is ffloating.
    betpeq datatype is ffloating.
    bettdi datatype is ffloating.
    betteq datatype is ffloating.
    tauedi datatype is ffloating.
    taueeq datatype is ffloating.
end magset relation.
! INDEX for MAGSET
define index magset_shotnum for magset
    duplicates are allowed.
    shotnum.
end magset_shotnum index.
commit
exit
$exit

```

Fig. 5.5. RDO commands for defining MAGSET.

.....

into a sequential file and then defining the index), it was seen that for MAGSET there was little difference in system usage.

During database definition, two files are created: MFE.Rdb (890 blocks) and MFE.SNP (202 blocks). The file with extension Rdb contains the database which consists of one relation for each user-defined relation and ten Rdb-defined relations comprising Rdb's data dictionary. Before any user data are loaded, the database occupies 890 blocks; this is Rdb overhead and in some cases may be much larger than the actual data.

The SNP file is a snapshot file used in READ ONLY transactions. If such transactions are large, this file may grow to thousands of blocks; and although the space is reused during subsequent transactions, it is not released.

The RDO command ANALYZE is used to examine the database. Figures 5.6 and 5.7, displays produced by the ANALYZE command, show information about MFERdb after database definition and before storing user data.

```

.....
Record Name              Occurrences  Bytes Used  Avg Rec in Bytes  % Frag  % Total Space  % Used Space
-----
MAGSET                   0             0           0             0       0       0
RDB$CONSTRAINTS         0             0           0             0       0       0
RDB$CONSTRAINT_RELATIONS 0             0           0             0       0       0
RDB$DATABASE            1             597         597           0       11      17
RDB$FIELDS              68           27404        403           0       72      69
RDB$FIELD_VERSIONS     104           8528         82           0       58      70
RDB$INDEX_SEGMENTS     20            1440         72           0       25      32
RDB$INDICES             15            1425         95           0       25      32
RDB$RELATIONS           11            968          88           0       17      25
RDB$RELATION_FIELDS    104           46904        451           0       84      93
RDB$VIEW_RELATIONS      0             0           0             0       0       0
-----
                          323           87266

```

Fig. 5.6. ANALYZE RELATIONS display after database definition.

```

.....
Space utilization analysis - completed at 31-MAY-1985 17:12:50.22
261 data pages, each page is 2 blocks long
Available data storage area is 70 percent utilized

-- %used --- #data pages -----
90 - 100%          66 |=====
80 - 90%           66 |=====
70 - 80%           34 |=====
60 - 70%           2  |=====
50 - 60%           4  |=====
40 - 50%           1  |=====
30 - 40%           43 |=====
20 - 30%           0  |=====
10 - 20%           0  |=====
0 - 10%            45 |=====

```

Fig. 5.7. ANALYZE PAGES display before storing user data.

In addition to the data dictionary within the Rdb database, metadata will also be stored in the Common Data Dictionary (CDD) if it is present on the VAX. The CDD is updated automatically upon initial database definition. In subsequent database usage, definition modifications are made

in the CDD only if the database is invoked with the CDD path specified. Care must be taken or inconsistencies may occur between data in the CDD and metadata within the Rdb database.

Defining must be done through RDO, but storage of data from RMS files into Rdb databases cannot be accomplished using RDO. RDO has no user interface with RMS; to read from or write to RMS files, either a host language program or DATATRIEVE must be used. At the time of Phase I, DATATRIEVE v3.0 (which interfaces with Rdb) was not available, so data were stored using Rdb from FORTRAN.

Using Rdb with a host language was quite easy. With few exceptions, the syntax is the same as that of RDO. The commands are embedded in the host language code with the addition of the Rdb statement flag, &Rdb&, in columns 1 through 5. (More discussion of the host language interface is found in Sect. 7.) Figure 5.8 is a listing of a portion of the FORTRAN program used in loading the relation MAGSET.

Use of a context variable may be unfamiliar to the reader. "M" in the STORE statement is an example of a context variable. Its use clearly indicates the relation to which a field belongs. Context variables may be used in DATATRIEVE and S1032 and on occasion are necessary in those systems. Such variables are required in the data manipulation language of both INGRES and Rdb.

### 5.2.3. S1032

The command file shown in Fig. 5.9 was used in defining and loading MAGSET in S1032. File MFE.DMB was created. For the version (3.0) of S1032 used in this study, it is as efficient in subsequent database usage for datasets to be defined separately, creating DMS files, as it is for all relations to be stored as a database in one DMB file.

S1032 is the only system with the recommendation to index while loading. It is somewhat more efficient to index (key) at load time than to store data into a sequential file and then index. In generating, maintaining, and using S1032, indexes were found to be very efficient.

The interactive LOAD command is the most efficient way to read data from an RMS file and store them in an S1032 dataset. The format of the data in an RMS file is described in a record descriptor using either the ATTRIBUTE or the RD command. As seen in Fig. 5.9, internal and external data formats can be described in the ATTRIBUTE command. There are occasions, however, when the RD command, which provides more flexibility, must be used.

### 5.2.4. Performance and Size

Table 5.1 shows the system usage statistics for defining and loading the MAGSET relation consisting of 827 records in the three database systems. The same relative performance was seen with the other two relations and with the larger MFE databases; S1032 did the task in 15-20% less time.

```

.....
      integer*4 shotnum
      real*4 lambda, lihaf, betpdi, betpeq, bettdi, betteq, tauedi, taupeq

&Rdb& INVOKE DATABASE FILENAME "rdb$db"

      open(unit=10, file='sys$userc:[kannan.data]magset.dat',
           1   status='old', err=9991)

&Rdb& START TRANSACTION READ WRITE RESERVING MAGSET FOR EXCLUSIVE WRITE

1000  continue

      read(10, 150, err=9992, end=9000) shotnum, lambda, lihaf, betpdi,
           1   betpeq, bettdi, betteq, tauedi, taupeq
150   format(i13, 8(f10.0))

&Rdb& STORE N IN MAGSET USING
&Rdb&      ON ERROR
           go to 9993

&Rdb&      END_ERROR
&Rdb&      N.SHOTNUM=shotnum;
&Rdb&      N.LAMBDA=lambda;
&Rdb&      N.LIHAF=lihaf;
&Rdb&      N.BETPDI=betpdi;
&Rdb&      N.BETPEQ=betpeq;
&Rdb&      N.BETTDI=bettdi;
&Rdb&      N.BETTEQ=betteq;
&Rdb&      N.TAUEDI=tauedi;
&Rdb&      N.TAUPEQ=taupeq;
&Rdb& END_STORE

           go to 1000

9000  continue
*
&Rdb& COMMIT

```

Fig. 5.8. Partial listing of FORTRAN program used to load MAGSET in Rdb.

```

.....

$!
$sl032

create database mfe output sys$sysdevice:[rdb.sl032]mfe

database mfe

dataset magset
attribute shotnum or sn integer keyed length 13
attribute lambda or lam real length 10
attribute lihaf real length 10
attribute betpdi real length 10
attribute betpeq real length 10
attribute bettdi real length 10
attribute betteq real length 10
attribute tauedi real length 10
attribute taueeq real length 10
end_dataset

end_database

load magset data_input sys$userc:[kannan.data]magset.dat

```

Fig. 5.9. Command file to define and load MAGSET in S1032.

.....

As seen in Table 5.2, S1032 databases are the smallest of the three. This advantage decreases with increasing database size. The larger data dictionary overhead associated with INGRES and Rdb databases is fixed to a certain extent, thus becoming less significant with larger databases.

### 5.3. OPERATIONS

In selecting the operations to use with the MFE database, relational operators were tested on experimental data. The selected 16 operations comprising Phase I performance evaluation appear on the following pages. Operations are described, syntax for each system given, and performance measurements for each of the three databases tabulated. Numbers 1, 2, and 3 in the following tables refer to the databases of 2,330, 23,300, and 93,200 records, respectively.

**Table 5.1. Loading and Indexing MAGSET relation**  
(827 records, 1 index)

System	Time <sup>a</sup>		Counts		
	Elapsed	CPU	Buf I/O	Dir I/O	Page fits
INGRES	:18	:16	182	162	785
Rdb	:18	:16	3	112	166
S1032	:16	:13	45	57	362

<sup>a</sup> Here and in the following tables, time is expressed in the format hh:mm:ss with leading zeroes omitted. Fractions of seconds are given when the measured time is less than one second.

**Table 5.2. Space usage of the MFE database**

System	Database size (disk blocks)		
	2330	23300	93200
INGRES	1000	6428	24432
Rdb	1682	8222	29824
S1032	696	5460	21852

The syntax shown is for interactive retrieval to the terminal (SYS\$OUTPUT). The performance figures for operations which retrieve a large number of records are for output to an RMS sequential disk file.

Operations for S1032 were run from command procedures containing S1032 interactive commands. INGRES operations also were executed from command procedures of interactive commands; however, statistics obtained referred only to the INGRES backend process. Even though the backend process accounts for most of the resource usage, collecting total usage data required embedding the commands in a host language and using system service routines to gather statistics on both processes. The tabulated numbers for INGRES are the total for both processes and are for use of INGRES from FORTRAN. The large buffered I/O counts are incurred because of data being passed by mailbox from the INGRES backend process to the FORTRAN program. Displaying data during an interactive INGRES session uses the same mechanism. Retrieving into another relation requires no transfer of data between the two INGRES processes; neither does the interactive reading or writing of an RMS file. For such operations, buffered I/O is negligible.

All operations were expressed in RDO syntax and run, but RDO is not a practical tool for evaluating performance. Within RDO, the only way to determine system usage is with control-T. Because there is no RMS interface, all output must be to the terminal or into a relation. After testing with RDO, the commands were embedded in a FORTRAN program and all performance measurements obtained from execution of the program.

For the most part, the following query syntax and tabulated performance statistics require no explanation. Comments are included as footnotes where clarification is needed. Tabulated data show the relative performance of the three systems and the relationship of performance to database size within a system. As expected, an increase in database size results in a corresponding increase in response time (and CPU time). However, in most cases the ratio of response times was somewhat less than the ratio of numbers of records in the database. There was no significant difference among the three products in this respect.

Overall, Rdb exhibited the best performance. For operations on one relation, all three systems were comparable. In the case of global aggregate operations (9, 10, 11), S1032 did better than Rdb, which in turn performed much better than INGRES. Operations 8, 12, 14, 15, 16 test the join operation capability. Because S1032 does not yet support a join operation, its performance was significantly poorer than INGRES and Rdb. Figure 5.10 graphically illustrates these results for the database of 93,200 records.

## OPERATION 1

An operation on one relation  
 Selecting (projecting) three fields from whole relation

INGRES: range of s is sunset<sup>a</sup>  
 retrieve (s.shotnum, s.ip, s.pheat)  
 \g

Rdb: for s in sunset  
 print s.shotnum, s.ip, s.pheat  
 end\_for

S1032: set sunset  
 find all  
 print shotnum, ip, pheat

		<u>Time</u>		<u>Counts</u>		
		<u>Elapsed</u>	<u>CPU</u>	<u>Buf I/O</u>	<u>Dir I/O</u>	<u>Page fits</u>
INGRES	1	:07	:06	195	96	260
	2	1:01	:55	1882	959	354
	3	4:22	3:42	7505	3848	348
Rdb	1	:08	:06	9	90	527
	2	1:02	:51	14	755	513
	3	4:05	3:24	35	2963	507
S1032	1	:10	:08	4	127	145
	2	1:23	1:12	13	1226	166
	3	5:26	4:44	40	4893	143

---

<sup>a</sup>“Range” statements need to be made only once per INGRES session and not with each query as shown here for clarity.

## OPERATION 2

(Same as Operation 1 except selecting 17% of the records)  
 An operation on one relation  
 Selecting three fields from 17% of the records based on an  
 indexed field

INGRES: range of s is sunset  
 retrieve (s.shotnum, s.ip, s.pheat) where s.nbeams=1  
 \g

Rdb: for s in sunset with s.nbeams=1  
 print s.shotnum, s.ip, s.pheat  
 end.for

S1032: set sunset  
 find nbeams eq 1  
 print shotnum, ip, pheat

		Time		Counts		
		Elapsed	CPU	Buf I/O	Dir I/O	Page fts
INGRES	1	:03	:02	44	54	419
	2	:16	:14	327	434	436
	3	:59	:53	1272	1766	444
Rdb	1	:02	:01	1	47	21
	2	:15	:10	2	382	71
	3	1:03	:42	5	1652	36
S1032	1	:03	:02	4	52	77
	2	:20	:16	5	478	75
	3	1:19	:58	10	1900	89

## OPERATION 3

(Same as Operation 1 except selecting 83% of the records)  
 An operation on one relation  
 Selecting three fields from 83% of the records based on an  
 indexed field

INGRES: range of s is sunset  
 retrieve (s.shotnum, s.ip, s.pheat) where s.nbeams !=1  
 \g

Rdb: for s in sunset with s.nbeams { }1  
 print s.shotnum, s.ip, s.pheat  
 end\_for

S1032: set sunset  
 find nbeams ne 1  
 print shotnum, ip, pheat

		Time		Counts		
		Elapsed	CPU	Buf I/O	Dir I/O	Page flts
INGRES	1	:06	:05	163	96	424
	2	:57	:52	1566	950	404
	3	4:59	3:30	6246	3819	402
Rdb	1	:05	:04	1	75	15
	2	:55	:45	6	732	16
	3	3:40	3:01	24	2919	17
S1032	1	:08	:07	4	119	18
	2	1:13	1:01	11	1178	18
	3	4:48	4:03	34	4704	32

## OPERATION 4

An operation on one relation  
Minimum of an indexed field<sup>a</sup>

INGRES: range of s is sumset  
retrieve (minip==min(s.ip))  
  \g

Rdb: print min s.ip of s in sumset  
Alternative syntax (1): print min s.ip of s in sumset with s.ip > 0  
Alternative syntax (2): for first 1 s in sumset sorted by s.ip  
  print s.ip  
  end\_for

S1032: set sumset  
  find all  
  sort ip  
  getrecord  
  print ip  
  } print \$min(ip)

---

<sup>a</sup>All systems have the five aggregate functions: minimum, maximum, average, total, and count. At this point in their development, it appears that indexes are not used efficiently with aggregate functions. With Rdb and S1032, the examples below show that performance improves with alternative syntax.

		<u>Time</u>		<u>Counts</u>		
		<u>Elapsed</u>	<u>CPU</u>	<u>Buf I/O</u>	<u>Dir I/O</u>	<u>Page fits</u>
INGRES	1	:02	:02	7	93	427
	2	:18	:12	7	917	426
	3	1:11	:49	7	3677	426
Rdb	1	:02	:01	0	70	12
	1 <sup>a</sup>	:00.7	:00.5	0	16	10
	1 <sup>b</sup>	:00.03	:00.03	0	0	7
	2	:25	:12	0	697	18
	2 <sup>a</sup>	:03	:03	0	47	18
	2 <sup>b</sup>	:00.08	:00.04	0	2	10
	3	1:39	:49	0	2776	15
	3 <sup>a</sup>	:11	:09	0	122	53
	3 <sup>b</sup>	:00.1	:00.06	0	3	11
S1032	1	:01	:01	3	12	41
	1 <sup>c</sup>	:06	:05	0	112	16
	2	:02	:02	3	34	77
	2 <sup>c</sup>	:57	:39	0	1095	10
	3	:08	:07	3	118	319
	3 <sup>c</sup>	3:46	2:37	0	4377	12

---

<sup>a</sup>Use of alternative syntax (1).

<sup>b</sup>Use of alternative syntax (2).

<sup>c</sup>Use of aggregate function \$min.

## OPERATION 5

An operation on one relation  
 Minimum of an indexed field over 17% of the relation  
 selected on an indexed field

INGRES: range of s is sunset  
 retrieve (minip=min (s.ip where s.nbeams=1))  
 \g

Rdb: print min s.ip of s in sunset with s.nbeams=1

S1032: set sunset  
 find nbeams eq 1  
 sort ip  
 getrecord  
 print ip } print \$min(ip)

		Time		Counts		
		Elapsed	CPU	Buf I/O	Dir I/O	Page fts
INGRES	1	:02	:01	11	50	453
	2	:11	:06	11	422	451
	3	:43	:23	11	1734	453
Rdb	1	:01	:00.4	0	46	9
	2	:08	:03	0	374	11
	3	:34	:15	0	1623	10
S1032	1	:01	:01	3	1	11
	1 <sup>a</sup>	:02	:01	1	35	112
	2	:03	:02	3	34	19
	2 <sup>a</sup>	:15	:10	1	451	6
	3	:08	:06	3	117	43
	3 <sup>a</sup>	1:04	:37	1	1782	52

<sup>a</sup>Use of aggregate function \$min.

## OPERATION 6

An operation on one relation  
 Minimum of an indexed field over 83% of the relation  
 selected on an indexed field

INGRES: range of s is sunset  
 retrieve (minip==min(s.ip where s.nbeams !=1))  
 \g

Rdb: print min s.ip of s in sunset with s.nbeams ( )1

S1032: set sunset  
 find nbeams ne 1  
 sort ip  
 getrecord  
 print ip } print \$min(ip)

		Time		Counts		
		Elapsed	CPU	Buf I/O	Dir I/O	Page flts
INGRES	1	:04	:02	7	93	441
	2	:32	:16	7	916	433
	3	2:08	1:03	7	3677	439
Rdb	1	:03	:01	0	72	6
	2	:25	:13	0	698	7
	3	1:39	:51	0	2777	7
S1032	1	:01	:01	3	1	4
	1 <sup>a</sup>	:05	:04	1	110	2
	2	:03	:02	3	34	50
	2 <sup>a</sup>	:49	:35	1	1069	4
	3	:09	:07	3	118	133
	3 <sup>a</sup>	3:16	2:19	1	4271	8

<sup>a</sup>Use of aggregate function \$min.

## OPERATION 7

An operation on one relation  
Complex aggregate

INGRES: range of s is sumset  
 retrieve (minip= $\min(s.ip \text{ where } s.ip > \text{avg}(s.ip))$ )  
 \g

Rdb: print min s.ip of s in sumset with s.ip > average su.ip of su in sumset  
 Alternative syntax: print min s.ip of s in sumset with s.ip > 0 and s.ip > average su.ip  
 of su in sumset with su.ip > 0

S1032: set sumset  
 find all  
 variable x real  
 let x= $\$ave(ip)$   
 find ip gt x  
 sort ip  
 getrecord  
 print ip } print  $\$min(ip)$

		Time		Counts		
		Elapsed	CPU	Buf I/O	Dir I/O	Page fts
INGRES	1	:05	:03	7	177	443
	2	:46	:27	7	1823	442
	3	3:03	1:48	7	7345	441
Rdb	1	:05	:03	0	140	12
	1 <sup>a</sup>	:01	:01	0	16	12
	2	:50	:25	0	1394	18
	2 <sup>a</sup>	:07	:06	0	95	18
	3	3:18	1:40	0	5553	13
	3 <sup>a</sup>	:24	:21	0	244	13
S1032	1	:05	:04	3	113	63
	1 <sup>b</sup>	:08	:06	1	188	116
	2	:43	:31	3	1129	63
	2 <sup>b</sup>	1:11	:50	1	1868	305
	3	2:49	2:03	3	4552	254
	3 <sup>b</sup>	4:39	3:18	1	7519	415

---

<sup>a</sup>Use of alternative query syntax.

<sup>b</sup>Use of aggregate function \$min.

## OPERATION 8

A join of two relations on an indexed field  
 Selecting (projecting) one field from each relation

```

INGRES:  range of s is sumset
         range of n is nbiset
         retrieve (s.shotnum, n.nbeams) where s.shotnum==n.shotnum
         \g

Rdb:     for s in sumset cross n in nbiset over shotnum
         print s.shotnum, n.nbeams
         end_for

S1032:a  set sumset
         find all
         for each record do
         map to nbiset via shotnum
         for each record do
         write sumset.shotnum, nbiset.nbeams
         end_for
         end_for
  
```

---

<sup>a</sup>The `print` command in S1032 (v3.0) can be used only with fields from one relation. To output fields from different relations, "WRITE" must be used in single record mode.

		<u>Time</u>		<u>Counts</u>		
		<u>Elapsed</u>	<u>CPU</u>	<u>Buf I/O</u>	<u>Dir I/O</u>	<u>Page fts</u>
INGRES	1	:09	:07	105	130	631
	2	1:25	1:11	942	1227	1295
	3	6:26	5:28	3764	4976	1697
Rdb	1	:05	:04	1	50	110
	2	:47	:42	4	267	760
	3	3:27	2:49	24	1068	6201
S1032	1	:52	:47	2	170	185
	2	10:03	9:22	7	1644	170
	3	38:32	34:41	21	6590	295

## OPERATION 9

A global aggregate<sup>a</sup>  
 Selecting 17% of relation 1 based on an indexed field  
 An aggregate function on relation 2

INGRES: range of s is sumset  
 range of n is nbiset  
 retrieve (num=count (s.shotnum where s.shotnum=n.shotnum and n.nbeams=1))  
 \g

Rdb: print count of s in sumset cross n in nbiset over shotnum with n.beams=1

S1032: set nbiset  
 find nbeams eq 1  
 map to sumset via shotnum  
 print \$count

		Time		Counts		
		Elapsed	CPU	Buf I/O	Dir I/O	Page fits
INGRES	1	:06	:04	19	95	1039
	2	:40	:28	19	688	1047
	3	2:45	1:48	20	2620	1099
Rdb	1	:01	:01	0	5	11
	2	:13	:10	0	211	181
	3	:32	:30	0	921	149
S1032	1	:02	:01	0	18	54
	2	:08	:07	0	234	218
	3	:30	:26	0	974	99

<sup>a</sup>“Global aggregate” refers to the evaluation of an aggregate or statistical function on records in one relation grouped according to a field in another relation. It is implemented in very different ways in these products: INGRES first forms a cartesian product of the two relations; Rdb uses nested loops; and S1032 here uses only indexes.

## OPERATION 10

A global aggregate  
 Selecting 83% of relation 1 based on an indexed field  
 An aggregate function on relation 2

INGRES: range of s is sunset  
 range of n is nbiset  
 retrieve (num=count (s.shotnum where s.shotnum=n.shotnum and n.beams !=1))  
 \g

Rdb: print count of s in sunset cross n in nbiset over shotnum with n.beams { } 1

S1032: set nbiset  
 find nbeams ne 1  
 map to sunset via shotnum  
 print \$count

		Time		Counts		
		Elapsed	CPU	Buf I/O	Dir I/O	Page flts
INGRES	1	:10	:07	17	145	938
	2	1:34	1:16	32	1288	1591
	3	8:36	5:15	64	5142	2302
Rdb	1	:02	:02	0	37	17
	2	:21	:17	0	235	620
	3	1:38	1:12	8	963	4676
S1032	1	:01	:01	0	7	19
	2	:10	:10	0	137	23
	3	:51	:39	0	544	24

## OPERATION 11

A global aggregate  
 Selecting 40% of relation 1 based on an indexed field  
 An aggregate function on relation 2

INGRES: range of s is sunset  
 range of n is nbiset  
 retrieve (num=count (n.shotnum where n.shotnum=s.shotnum and s.ip >=200.0 and s.ip <=300.0))  
 \g

Rdb: print count of n in nbiset cross s in sunset over shotnum with s.ip between 200.0 and 300.0

S1032: set sunset  
 find ip between 200.0 and 300.0  
 map to nbiset via shotnum  
 print \$count

		Time		Counts		
		Elapsed	CPU	Buf I/O	Dir I/O	Page fits
INGRES	1	:09	:06	17	147	937
	2	1:15	:55	18	1246	1543
	3	5:18	3:59	54	5018	2302
Rdb	1	:06	:03	0	203	24
	2	1:17	:23	9	3162	276
	3	5:30	2:00	10	13313	527
S1032	1	:02	:02	0	19	13
	2	:15	:14	0	253	20
	3	1:00	:56	0	1040	69

OPERATION 12

A join of two relations on an indexed field  
Selecting 40% of relation 1 based on an indexed field  
Selecting (projecting) one field from each relation

INGRES: range of s is sumset  
range of n is nbiset  
retrieve (n.shotnum, s.ip) where n.shotnum=s.shotnum and s.ip >= 200.0 and s.ip <= 300.0  
\\g

Rdb: for s in sumset cross n in nbiset over shotnum with s.ip between 200.0 and 300.0  
print n.shotnum, s.ip  
end\_for

S1032: set sumset  
find ip between 200.0 and 300.0  
for each record do  
map to nbiset via shotnum  
for each record do  
write nbiset.shotnum,ip  
end\_for  
end\_for

		<u>Time</u>		<u>Counts</u>		
		<u>Elapsed</u>	<u>CPU</u>	<u>Buf I/O</u>	<u>Dir I/O</u>	<u>Page fts</u>
INGRES	1	:08	:06	55	129	656
	2	1:15	:55	458	1214	1033
	3	5:22	4:04	1797	4917	1989
Rdb	1	:07	:04	1	202	71
	2	1:28	:40	11	3171	244
	3	6:13	2:42	17	13353	117
S1032	1	:23	:20	2	121	45
	2	4:27	3:58	3	1266	58
	3	17:18	14:35	10	5072	89

## OPERATION 13

A global aggregate<sup>a</sup>  
 Project on an indexed field  
 Aggregate function on relation 2

INGRES: range of s is sumset  
 range of n is nbiset  
 retrieve (n.nbeams, num=count(s.shotnum by n.nbeams  
 where s.shotnum=n.shotnum))  
 \g

Rdb: for n in nbiset reduced to n.nbeams  
 print n.nbeams, count of s in sumset with s.nbeams=n.nbeams  
 end\_for

S1032: set nbiset  
 find all  
 map to sumset via shotnum  
 values nbeams

		Time		Counts		
		Elapsed	CPU	Buf I/O	Dir I/O	Page fits
INGRES	1	:19	:13	38	231	1264
	2	2:40	2:11	79	1870	2316
	3	10:23	8:43	118	7259	3417
Rdb	1	:01	:01	0	15	17
	2	:08	:07	0	34	131
	3	:36	:33	0	194	12
S1032	1	:02	:01	0	16	29
	2	:12	:11	0	240	55
	3	:55	:42	0	1000	192

<sup>a</sup>Because of the content of this database, the join may be on either field shotnum or nbeams.

## OPERATION 14

A join of all three relations in the database  
 Selecting 20% of relation 1 based on an indexed field  
 Selecting (projecting) fields from relations 2 and 3

INGRES: range of s is sumset  
 range of n is nbiset  
 range of m is magset  
 retrieve (s.shotnum, m.lambda) where s.shotnum=n.shotnum  
 and s.shotnum=m.shotnum and n.nbeams=1  
 \g

Rdb: for n in nbiset cross s in sumset cross m in magset  
 with s.shotnum=n.shotnum and s.shotnum=m.shotnum  
 and n.nbeams=1  
 print s.shotnum, m.lambda  
 end\_for

S1032: set nbiset  
 find nbeams eq 1  
 map to sumset via shotnum  
 for each record do  
 map to magset via shotnum  
 for each record do  
 write sumset.shotnum, lambda  
 end\_for  
 end\_for

		<u>Time</u>		<u>Counts</u>		
		<u>Elapsed</u>	<u>CPU</u>	<u>Buf I/O</u>	<u>Dir I/O</u>	<u>Page flts</u>
INGRES	1	:08	:06	36	94	680
	2	1:04	:51	239	835	743
	3	3:32	2:34	856	3346	1014
Rdb	1	:03	:02	1	46	263
	2	:33	:24	1	403	2399
	3	1:35	1:23	4	862	3385
S1032	1	:15	:12	2	104	53
	2	3:18	2:42	2	1167	79
	3	8:28	6:43	5	4010	192

## OPERATION 15

(Same as Operation 14 except 4 times the number of records.)

		Time		Counts		
		<u>Elapsed</u>	<u>CPU</u>	<u>Buf I/O</u>	<u>Dir I/O</u>	<u>Page fts</u>
INGRES	1	:12	:10	106	148	664
	2	1:58	1:38	936	1438	1585
	3	8:37	7:27	3599	5799	3945
Rdb	1	:06	:05	1	61	48
	2	1:00	:52	4	522	800
	3	4:13	3:26	22	2098	4471
S1032	1	:42	:37	2	68	28
	2	7:40	6:57	5	1623	0
	3	28:22	25:54	17	6541	108

## OPERATION 16

A join of all three relations  
 Selecting 40% of relation 1 based on an indexed field  
 Selecting (projecting) fields from relations 2 and 3

INGRES: range of s is sumset  
 range of n is nbiset  
 range of m is magset  
 retrieve (n.shotnum, m.lambda) where n.shotnum=m.shotnum  
 and s.shotnum=n.shotnum  
 and s.ip >= 200.0 and s.ip <= 300.0

\g

Rdb: for s in sumset cross n in nbiset cross m in magset  
 with s.ip between 200.0 and 300.0  
 and n.shotnum=m.shotnum and s.shotnum=n.shotnum  
 print n.shotnum, m.lambda  
 end\_for

S1032: set sumset  
 find ip between 200.0 and 300.0  
 map to nbiset via shotnum  
 for each record do  
 map to magset via shotnum  
 for each record do  
 write nbiset.shotnum, lambda  
 end\_for  
 end\_for

		<u>Time</u>		<u>Counts</u>		
		<u>Elapsed</u>	<u>CPU</u>	<u>Buf I/O</u>	<u>Dir I/O</u>	<u>Page fits</u>
INGRES	1	:12	:09	84	152	733
	2	1:57	1:33	727	1437	1631
	3	3:30	6:52	2715	5775	4086
Rdb	1	:08	:04	9	179	30
	2	1:43	:50	11	3438	454
	3	6:55	3:24	17	14406	849
S1032	1	:24	:21	2	144	301
	2	4:01	3:27	3	1309	363
	3	15:50	13:52	9	5277	597

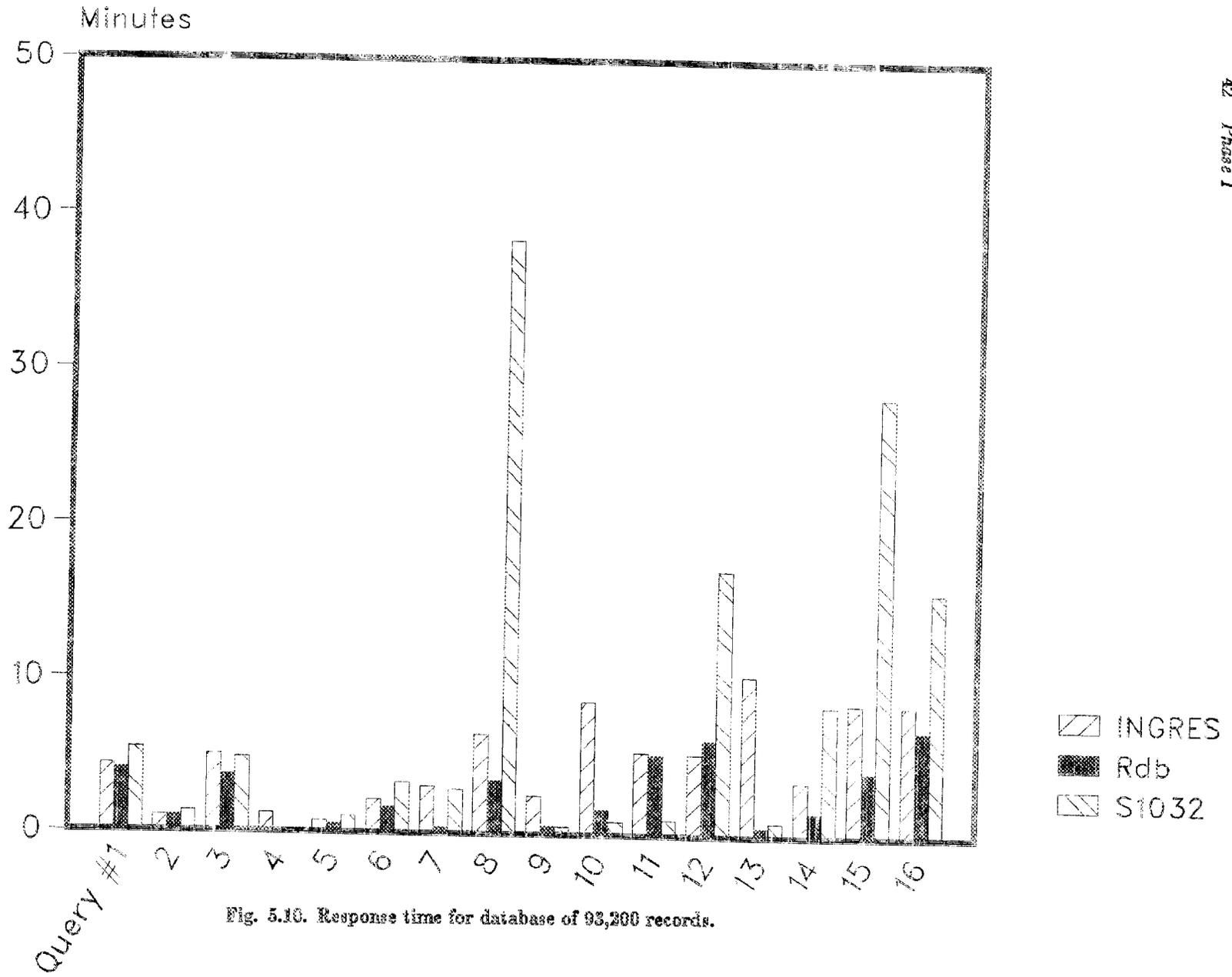


Fig. 5.10. Response time for database of 93,200 records.

## 6. PHASE II

The goal of Phase II was to evaluate database systems for the Energy Systems' VAX environment. Although large mainframe processors retain their importance, there is a growing number of users of VAX-class machines, in particular, the VAX itself. Whether new to computers or migrating from larger machines such as IBM mainframes and the PDP-10 or from smaller machines of the PDP-11 class, users appreciate the ease of use of the VAX/VMS and are interested in software products that have this characteristic. In the database management field, that implies products based on the relational model.

The products included in this phase of the evaluation were DATATRIEVE, INGRES, Rdb, and S1032. INGRES and Rdb were of particular interest because of their full relational functionality, S1032 because of its syntactical similarity to S1022 and the long relationship and good reputation Software House has with Energy Systems, and DATATRIEVE because it was already installed on many Energy Systems' VAXs. There was also interest in Rdb and DATATRIEVE because of their compatibility with existing and proposed DEC products. Other relational database systems were considered but were not included for a variety of reasons such as lack of time, unavailability for on-site evaluation, cost, and limited relational functionality.

Many potential users of a VAX database management system are accustomed to S1022 and its capabilities. They expect and need these capabilities in their VAX work. An effort was made to include data and operations common to S1022 in this phase of the evaluation.

The database was defined and loaded, and all operations were executed in S1022 on the PDP-10 of the Fusion Energy Division's (FED) User Service Center. The FED computer operations staff was very helpful; all user jobs and network file transfers were stopped in order to simulate the single-user environment of the VAX tests.

Accounting methods on the PDP-10 and VAX differ in the way system usage is attributed to a user process. S1022 statistics are given along with those of the VAX systems; however, without knowing the system accounting algorithms, it is difficult to compare performance figures other than elapsed time. That is an interesting comparison, particularly to the end user, because it indicates the difference in response time likely to be experienced.

### 6.1. DATABASE DESIGN

Figure 6.1 shows the composition of the database, TESTDB, used in Phase II. The relations PHOTO and TITLE were S1022 datasets. They were written from S1022 to sequential disk files, copied to the VAX using DECnet, and then easily defined, preserving datatypes and indexing of S1022, and loaded in the four systems.

---

<b>TESTDB</b>		
4 RELATIONS		
PHOTO	5000	RECORDS 23 FIELDS; 185 BYTES; 12 KEYS 11 TEXT 8 INTEGER 4 DATE
TITL	10112	RECORDS 3 FIELDS; 48 BYTES; 3 KEYS 1 TEXT 2 INTEGER
DIVNAM	137	RECORDS 2 FIELDS; 39 BYTES; 1 KEY 2 TEXT
XPRANT	8270	RECORDS 15 FIELDS; 68 BYTES; 3 KEYS 2 TEXT 7 INTEGER 3 REAL 1 REAL DOUBLE PRECISION

**Fig. 6.1. Composition of TESTDB.**

---

## 6.2 DATABASE DEFINITION AND STORAGE

Performance statistics for defining, loading, and indexing TESTDB are given in Table 6.1 along with the space used by the resulting database.

Performance numbers for S1022 are given for completeness, but they should be used with caution. For many operations, the observed elapsed time for S1022 was close to that of the VAX systems; however, the CPU time was generally much less than for any of the VAX systems. It was not determined how much of the difference is attributable to system accounting. The PDP-10 does

**Table 6.1. Defining, loading, and indexing TESTDB  
(4 Relations)**

System	Time		Counts			Disk
	Elapsed	CPU	Buf I/O	Dir I/O	Page flts	Blocks
DTR <sup>a</sup>	27:03	19:42	577	19448	8585	11030
INGRES	20:21	12:55	4789	13415	10331	6628
Rdb	19:28	13:54	316	9392	7967	8827
S1032	11:35	10:08	741	4264	5650	5631
			<b>READ</b>	<b>WRITE</b>		
S1022	9:18	3:40	13348	13712		4727

<sup>a</sup>In tables and figures DATATRIEVE will be abbreviated to DTR.

not differentiate between the buffered and direct I/O; I/O statistics are block reads and writes, as indicated.

Taking the PDP-10 word size of 36 bits into account, it is seen that the S1022 database size in bytes is very close to that of S1032. A PDP-10 word contains four 8-bit bytes or five 7-bit bytes. S1022 stores text as 5 bytes per word. A block consists of 128 words or 640 bytes of text compared to the VAX's 512 bytes per block.

Indexing the PHOTO relation (12 keys) accounted for approximately one-half the elapsed time for INGRES and Rdb. So many indexes are unlikely in these two systems, but the design of TESTDB preserved S1022 indexes. Indexes are not as essential to INGRES and Rdb as they are to DATATRIEVE and S1032. No operations in the former two require indexes; performance may be improved with the use of indexes, but they are never necessary. With S1032, data manipulation language and operations are dependent on data structure; indexes are required in some cases. Indexes are also necessary for some operations in DATATRIEVE.

The numbers for DATATRIEVE in Table 6.1 resulted from restructuring the data in DATATRIEVE and then using the RMS utility procedure CONVERT for indexing. Table 6.2 contrasts indexing by CONVERT with indexing by DATATRIEVE for the two relations PHOTO and XPRMNT. In all cases, data were first restructured by DATATRIEVE. This included converting some ASCII fields to numeric and eliminating some fields. Table 6.2 makes clear the important point that DATATRIEVE is not a database management system. It is a query language and report writer and should be used for its intended purpose; it interfaces with RMS, Rdb, and DBMS. Databases should be maintained and tuned by the system in which they reside.

**Table 6.2. Indexing by DATATRIEVE and CONVERT**

Relation	Time		Counts		
	Elapsed	CPU	Buf I/O	Dir I/O	Page fits
PHOTO					
DTR & CONVERT	10:40	7:54	316	5541	4425
DTR	1:46:31	24:06	1562	334535	516
XPRMNT					
DTR & CONVERT	11:10	8:30	102	9910	1390
DTR	52:10	17:31	1051	120123	514

The impact upon resource usage does not end with the initial load. Databases indexed by DATATRIEVE generally require more space (44% more for XPRMNT) and perform less efficiently in subsequent usage, incurring larger direct I/O counts because of DATATRIEVE's bucket size of two blocks.

### 6.3. OPERATIONS

A set of 26 operations to be used with TESTDB was defined. The following types of operations were included; select, project, join, aggregate, sort, append, delete, modify, and dynamic database restructuring. Results from select and project operations were written to (1) the screen, (2) RMS files, and (3) database relations. Where applicable, operations were repeated for 1%, 10%, and 20% of a relation. Operations were performed on the indexed database; indexes were deleted; then operations were repeated, where possible, on the sequential database.

All operations were executed from the interactive language of each system. This was in keeping with the Phase II goal of evaluation on the basis of common S1022 usage. Recorded results are for the INGRES backend process; however, experience with Phase I ensures that for types of operations included in Phase II the INGRES frontend process contributes only negligibly to the system usage statistics. DATATRIEVE was used with both RMS and Rdb. In addition, Rdb operations, where possible, were executed using RDO.

A representative sample of the 26 operations is presented in the sections that follow. Discussion also includes operations not shown.

#### 6.3.1. Selection and Projection

The relation PHOTO was used for select and project operations. Selection was based on the value of an indexed field. Four fields (52 bytes) were projected and written to the terminal

(SYS\$OUTPUT). The operation was repeated for output to an RMS file and then again for output to a relation. The sequence was executed for selection of ~1% (49 records) of the relation. When the sequence was repeated for 10% (513 records) and ~20% (1072 records) of the relation, output was only to RMS files and relations; there were too many records retrieved for useful terminal display. For 1% of the relation, performance for retrieval to the terminal was close to that for retrieval to an RMS file. Figure 6.2 shows the syntax for retrieval to an RMS file and Fig. 6.3 the syntax for retrieval to a relation. Tables 6.3 and 6.4 show the corresponding performance. Selection was determined by the field "div," which was a keyed field. The index on "div" was deleted and the operations repeated. For DATATRIEVE and INGRES, where there are primary and secondary indexes, the tests were run with "div" as the primary index and then again with "div" as a secondary index. These conditions are indicated in the tables.

---

	An operation on one relation (5000 records) Selection of 20% of the relation based on an indexed field Projection of 4 fields (32 bytes) to an RMS file
DTR with RMS	For PHOTO with div starting with "Y" - print name, addr, acct, wo on DIVY.LIS
INGRES	range of p is PHOTO retrieve into DIVY (p.name, p.addr, p.acct, p.wo) where p.div="Y*" copy DIVY (name=c0, addr=c0, acct=c0, wo=c0nl) into DIVY.LIS
Rdb/RDO	RMS files cannot be read or written from RDO
Rdb/DTR	Same syntax as DTR with RMS
S1032	set ds PHOTO find div beg Y initialize 3 DIVY.LIS print on 3 name, addr, acct, wo release 3

---

Fig. 6.2. Syntax for retrieval to an RMS file.

Some basic differences among the four systems are pointed out in these operations. Forming a relation is a natural operation in INGRES. Field descriptions are assumed from the existing relations in which the fields reside. For the other systems, a relation must be explicitly defined prior to storing data in it. With Rdb, this removes some spontaneity from database activity; if using an Rdb database from DATATRIEVE, the user cannot retain retrieved information in an Rdb relation unless the relation had been anticipated and previously defined using RDO.

```

.....

                An operation on one relation (5000 records)
                Selection of 20% of the relation based on an indexed field
                Projection of 4 fields (32 bytes) to a relation

D/TR with RMS  Define domain DIVY using DIVY.REC on DIVY.DAT
                Define record DIVY_REC using
                  01 DIVY_REC.
                    05 name pic x(25).
                    05 addr pic x(15).
                    05 acct usage is long.
                    05 wo pic x(8).
                ;
                Define file DIVY;
                Ready DIVY write
                DIVY=PHOTO with div starting with "Y"

INGRES         range of p is photo
                retrieve into divy (p.name, p.addr, p.acct, p.wo) where p.div="Y*"
                \g

Rdb/RDO        define relation DIVY.
                name datatype is text of size 25.
                addr datatype is text of size 15.
                acct datatype is signed longword.
                wo datatype is text of size 8.
                end DIVY relation.
                for p in PHOTO with p.div starting with "Y"
                store d in DIVY using
                  d.name=p.name;
                  d.addr=p.addr;
                  d.acct=p.acct;
                  d.wo=p.wo;
                end_store
                end_for

Rdb/DTR        Following definition of DIVY from RDO:
                For PHOTO with div starting with "Y"
                store divy using
                  begin
                    name=name
                    addr=addr
                    acct=acct
                    wo=wo
                  end

S1032         set ds PHOTO
                find div beg Y
                create ds DIVY
                attribute name text 25
                attribute addr text 15
                attribute acct integer
                attribute wo text 8
                end_dataset
                set dataset photo
                dump ds_output DIVY

```

Fig. 6.3. Syntax for retrieval to a relation.

Table 6.3. Retrieval to a relation

System	Time		Counts		
	Elapsed	CPU	Buf I/O	Dir I/O	Page fts
DTR with RMS	:13 <sup>a</sup>	:09	11	144	12
	:26 <sup>b</sup>	:13	11	872	9
	:19 <sup>c</sup>	:13	11	178	7
INGRES	:12 <sup>a</sup>	:06	63	146	436
	:29 <sup>b</sup>	:13	64	879	561
	:20 <sup>c</sup>	:11	61	436	551
Rdb/RDO <sup>d</sup>	:30	:16		983	270
	:36 <sup>c</sup>	:17		644	161
Rdb/DTR	:46	:29	26	1044	224
	:40 <sup>c</sup>	:30	18	499	133
S1032	:17	:11	18	491	46
	:45 <sup>c</sup>	:32	18	981	611
S1022	:08	:01	<b>READ</b> 795	<b>WRITE</b> 106	

<sup>a</sup>Selecting on a primary index.

<sup>b</sup>Selecting on secondary index.

<sup>c</sup>Selecting on an unindexed field.

<sup>d</sup>Statistics from using control-T; only one I/O figure given.

Writing selected information to an RMS file is a two-step process in INGRES. The information must first be written to a relation. In RDO it cannot be done at all; RMS files cannot be read or written from RDO.

The different treatment of indexes was alluded to in Sect. 2. In Rdb and S1032, there is no concept of primary versus secondary indexes; all indexes are created equal. In DATATRIEVE and INGRES with ISAM relations, the primary index is a clustering index and has an edge on performance over any secondary indexes. This is sharply illustrated in Tables 6.3 and 6.4. It is interesting to note that response time is significantly better for selection based on an unkeyed field than on a field having a secondary index defined. In the latter case, the increased buffered I/O

Table 6.4. Retrieval to an RMS file

System	Time		Counts		
	Elapsed	CPU	Buf I/O	Dir I/O	Page fts
DTR with RMS	:12 <sup>a</sup>	:09	4	89	31
	:25 <sup>b</sup>	:13	4	816	21
	:19 <sup>c</sup>	:14	4	123	22
INGRES	:19 <sup>a</sup>	:09	76	173	697
	:35 <sup>b</sup>	:16	77	906	814
	:26 <sup>c</sup>	:15	74	463	824
Rdb/DTR	:27 <sup>d</sup>	:16	4	815	32
	:22 <sup>c</sup>	:17	4	345	17
S1032	:16 <sup>d</sup>	:11	4	436	116
	:41 <sup>c</sup>	:32	4	860	31
S1022	:12	:04	793	94	

<sup>a</sup>Selecting on primary index.

<sup>b</sup>Selecting on secondary index.

<sup>c</sup>Selecting on an unindexed field.

<sup>d</sup>Selecting on an indexed field.

counts are incurred because the secondary index is searched as well as the base relation, which is not ordered by the indexed field.

DATATRIEVE does not attempt query optimization; it executes what the user prescribes. INGRES does have a query optimizer; however, here is an operation it does not seem to optimize. Retrieving 20% of a 5000-record relation based on a field which has a secondary index defined, the above tables indicate optimization should result in disregard of the index.

This pattern of a primary index being better than no index better than secondary index was not seen in selection of 1% and 10% of the relation. As shown in Table 6.5, the primary index always excelled, but the secondary index gave much better response time than the unkeyed field.

When DATATRIEVE is used for the selection of 20% of the Rdb relation, the index has a negative effect; however, Table 6.5 shows the benefit of the index when selecting fewer records. The greatest improvement from the use of indexes is seen in S1032, where the introduction of an

index results in decreases in response time of 82%, 71%, and 62% for 1%, 10%, and 20% retrieval, respectively.

**Table 6.5. Retrieval to a relation**  
(Response time)

System		Percentage of Relation Selected		
		1%	10%	20%
DTR with RMS	<sup>a</sup>	:04	:07	:13
	<sup>b</sup>	:05	:12	:26
	<sup>c</sup>	:12	:14	:19
INGRES	<sup>a</sup>	:04	:08	:12
	<sup>b</sup>	:06	:14	:29
	<sup>c</sup>	:18	:19	:20
Rdb/DTR	<sup>d</sup>	:04	:19	:46
	<sup>c</sup>	:16	:26	:40
S1032	<sup>d</sup>	:06	:12	:17
	<sup>c</sup>	:35	:41	:45

<sup>a</sup>Selecting on primary index.

<sup>b</sup>Selecting on secondary index.

<sup>c</sup>Selecting on an unindexed field.

<sup>d</sup>Selecting on an indexed field.

The relational operator "project" is commonly defined implying uniqueness. In DATATRIEVE and Rdb, the reduce command effects project with uniqueness. In addition, a sort order based on the ordering of the reduce fields is implicit in DATATRIEVE. In INGRES the all-inclusive "retrieve," with the qualifier "unique," projects output to the terminal but not to a relation; there is no "retrieve unique into." Projecting to a relation is accomplished by "retrieve into" with the default storage structure for relations set to hash, which means duplicate rows are removed. S1032 has no project command; a procedure with explicit looping is required. The same is true for S1022. These two systems show longer response times than the other systems. Although the response time for S1022 is 30% less than that of S1032, the CPU time is reduced by 84% presumably due to different

accounting algorithms for the PDP-10 and VAX. Figure 6.4 shows projection syntax and Table 6.6 the corresponding performance.

.....

Project (unique) 3 fields of one relation

DTR with RMS    Print PHOTO reduced to DIV, ACCT, SACCT on PROJECT.LIS

INGRES            range of p is PHOTO  
                   set ret into "hash"  
                   retrieve into proj (p.div, p.acct, p.sacct) sort by p.div, p.acct, p.sacct  
                   copy proj (div=c0, acct=c0, sacct=c0nl) into  
                   "sys\$userc:[Kannan.INGRES] PROJECT.LIS,text"  
                   \g

Rdb/RDO            (To a relation rather than an RMS file)  
                   for p in PHOTO reduced to p.div, p.acct, p.sacct  
                   sorted by p.div, p.acct, p.sacct  
                   store pr in proj using  
                   pr.div=p.div  
                   pr.acct=p.acct  
                   pr.sacct=p.sacct  
                   end\_store  
                   end\_for

Rdb/DTR            Same as DTR with RMS

S1032             set ds PHOTO  
                   init 3 PROJECT.LIS  
                   Begin  
                   Find all  
                   Sort div acct sacct  
                   Variable odiv Text 3 initially " "  
                   Variable oacct, osacct integer initially 0  
                   For each PHOTO record do  
                   If sacct ne osacct then  
                   write on 3 div, acct, sacct  
                   Let osacct=sacct, oacct=acct, odiv=div  
                   ELSE\_IF acct ne oacct then  
                   write on 3 div, acct, sacct  
                   Let oacct=acct, odiv=div  
                   ELSE\_IF div ne odiv then  
                   write on 3 div, acct, sacct  
                   Let odiv=div  
                   END\_IF  
                   END\_FOR  
                   END

Fig. 6.4. Projection syntax.

.....

Table 6.6. Projection

System	Time		Counts		
	Elapsed	CPU	Buf I/O	Dir I/O	Page fits
DTR with RMS	2:13	1:17	17	3866	280
INGRES	:34	:26	29	486	886
Rdb/RDO	:24	:18		400	852
Rdb/DTR	:24	:20	3	341	747
S1032	3:58	2:45	18	4739	640
			<b>READ</b>	<b>WRITE</b>	
S1022	2:46	:27	6676	300	

### 6.3.2. Joins

The join of most interest and use, the equijoin (based on equality of a common field), is presented in Sect. 5.3, Operations 8, 12, 14, 15, 16. Three of the 26 operations of Phase II are also equijoins. The order of performance was the same as in Phase I: Rdb (RDO and DTR), INGRES, S1032. [The version (3.0) of S1032 included in this investigation does not have a true join.] In terms of elapsed and CPU time, DATATRIEVE with RMS ISAM files showed significantly poorer performance than the other three systems. Join operations in S1022 were approximately 30% faster than in S1032, but they were somewhat slower than in Rdb and INGRES. In S1032 and S1022 indexed fields are necessary for a join. In DATATRIEVE, joining over unkeyed fields is possible but may be impractical; a join which took 2 minutes 50 seconds elapsed time for keyed fields in DATATRIEVE was stopped incomplete after 12 hours using unkeyed fields. This same join in Rdb over keyed fields took 50 seconds, and over unkeyed fields, 3 minutes 40 seconds. Whether or not the joining fields were keyed made virtually no difference to INGRES; the elapsed time of 1 minute 8 seconds occurred in both cases.

When an equijoin is performed, data are retrieved from the two relations only if the joining fields have matching values. Sometimes, however, in joining two relations it is useful to retrieve data from a relation whether or not a match is found. This type of join is called an outer join. Figure 6.5 and Table 6.7 summarize the outer join included in Phase II. Data were retrieved from PHOTO and, if corresponding data were present, also from DIVNAM.

There is much variety in the implementation of an outer join. RDO does not provide one. The other three systems handle it in very different ways: a view in DATATRIEVE, a procedure with looping in S1032, and delete, append, and intermediate relations in INGRES. The resulting RMS

```

.....

                                Outer JOIN

DTR with RMS      define domain view1 of photo, divnam
and Rdb           01 photo.flds occurs for photo.
                  05 name from photo.
                  05 phone from photo.
                  05 need from photo.
                  05 join.fld occurs for divnam with
                   photo_rec. div=divnum_rec.div.
                   09 divname from divnam.
;
ready view1
for view1 sorted by name print on
OUTERJOIN.LIS

INGRES            range of p is PHOTO
                  range of d is DIVNAM
                  set ret.into "heap"
                  retrieve into REL1 (p.div, p.name, p.phone, p.need, d.divname)
                   where p.div=d.div
                  retrieve into REL2 (p.div, p.name, p.phone, p.need)
                  range of r1 is REL1
                  range of r2 is REL2
                  delete r2 where r2.div=d.div
                  append to REL1 (r2.div, r2.name, r2.phone, r2.need)
                  retrieve into REL3 (r1.name, r1.phone, r1.need, r1.divname)
                   sort by r1.name
                  copy REL3 (name=c0, phone=c0, need=c0, divname=c0nl)
                   into "sys$userc:KANNAN.INGRES]OUTERJOIN.LIS,text"
\g

S1032            set ds PHOTO
                  init 3 OUTERJOIN.LIS
                  find all
                  sort name
                  for each PHOTO record do
                   map to DIVNAM via div
                   if $ nrec eq 0 then
                    write on 3 name, phone, need
                   else
                    write on 3, name, phone, need, divname
                   end.if
                  end.for

```

**Fig. 6.5. Syntax of outer JOIN**

.....

file contained 5000 records except from INGRES, where duplicate rows were eliminated in forming the relation REL3.

Table 6.7. Outer JOIN

System	Time		Counts		
	Elapsed	CPU	Buf I/O	Dir I/O	Page fts
DTR with RMS	5:27	2:05	36	6878	746
INGRES	3:48	2:32	996	2806	2999
Rdb/DTR	1:58	1:49	18	449	3763
S1032	7:36	6:31	16	4932	106
			<b>READ</b>	<b>WRITE</b>	
S1022	3:40	:45	6994	1020	

### 6.3.3. Maintenance Operations

Modifications of two kinds were made to the database: content and structure alterations. In the former case, data were modified in keyed and unkeyed fields for 1%, 10%, 20%, and 100% of a relation, and records were appended to and deleted from unkeyed and keyed relations. The results were not unexpected: maintenance on keyed fields uses more system resources.

The impact of index maintenance was minimal when modifying the value of an integer field for 1% of the PHOTO relation. Even when three fields in 20% of the PHOTO relation were modified, whether the fields were keyed or unkeyed made little difference to the performance of S1032; execution time was approximately 50 seconds in both cases. INGRES showed a six-fold increase in execution time for keyed fields from 20% of the relation: from 30 seconds to 3 minutes. With DATATRIEVE the degradation was even more pronounced: from 30 seconds to 4.5 minutes with RMS, from 50 seconds to 3.5 minutes with Rdb.

Table 6.8 shows the performance for the extreme case of field modification over an entire relation. The syntax for this operation is seen in Fig. 6.6. Because the index on the field *ip* required a few seconds to delete and less than a minute to build, it is obvious from Table 6.8 that the index should be deleted before the maintenance described here is performed.

The number of direct I/Os for DATATRIEVE was out of line with the other systems. This behavior was observed in two sorting operations (not shown) and in Table 6.2. Here is another occasion to use RMS facilities. There are RMS parameters to tune which could conceivably affect the I/O usage.

Other maintenance operations performed included appending and deleting records. These operations were performed on both indexed and unindexed relations except in the case of DATATRIEVE, where records cannot be deleted from an unindexed relation.

**Table 6.8. Modification of a field over whole relation**

System	Time		Counts		
	Elapsed	CPU	Buf I/O	Dir I/O	Page flts
DTR with RMS	26:40 <sup>a</sup>	9:49	0	74862	21
	1:32 <sup>b</sup>	1:21	321	12	
INGRES	9:50 <sup>a</sup>	9:31	89	26217	439
	1:30 <sup>b</sup>	1:05	46	1201	453
Rdb/DTR	14:08 <sup>a</sup>	6:57	503	21074	13290
	3:15 <sup>b</sup>	2:40	73	1296	1763
S1032	10:42 <sup>a</sup>	9:29	18	14862	317
	3:30 <sup>b</sup>	2:07	0	733	210
S1022	15:55 <sup>a</sup>	1:43	<b>READ</b> 22600	<b>WRITE</b> 15578	

<sup>a</sup>Modifying an indexed field.

<sup>b</sup>Modifying an unindexed field.

Deletion of records was based on a field value selection: all records in the relation PHOTO with div="K20." This resulted in the removal of 31 records, 0.6% of the relation. In INGRES and S1032, deletion was much faster on the indexed relations, presumably due to the initial search having more impact than index maintenance for such a small number of records. In S1022, this operation was instantaneous whether indexed or not. Rdb from DATATRIEVE required more time for the indexed relation than the unindexed. The execution time for DATATRIEVE with RMS was 30 seconds, which was significantly slower than the other systems. The direct I/O usage was also much greater.

The append operation consisted of adding one hundred records in an RMS sequential file to the relation XPRMNT. This operation could not be performed in Rdb from RDO because of the lack of an RMS interface.

The records were added to XPRMNT with three indexes defined and also to XPRMNT with all indexes deleted. Performance was significantly better for the unindexed relation. Execution time is shown in Table 6.9.

---

DTR with RMS and Rdb	ready XPRMNT write for XPRMNT modify using ip=ip*1.05
INGRES	range of x is XPRMNT replace x(ip=1.05*x.ip) \\g
Rdb/RDO	for x in XPRMNT modify x using x.ip=1.05*x.ip end_modify end_for
S1032	set ds XPRMNT find all change is (ip*1.05)

**Fig. 6.6. Modification of a field over whole relation.**

---

**Table 6.9. Append operation execution time**

	Execution time	
	Indexed	Unindexed
DTR with RMS	:40	:08
INGRES	:19	:08
Rdb/DTR	:41	:15
S1032	:17	:04
S1022	:08	:02

As for modifications to the structure of the database, relational systems proponents claim it can be done, and moreover, done easily and dynamically. Certainly indexes can be added and deleted at will; relations likewise with facility. (The latter is a matter of course with DATATRIEVE where each relation is a separate domain.) But what about altering structure internal to a relation, adding a

field, deleting a field, or changing the datatype of a field? Only Rdb (through RDO) can accomplish these feats without going through some amount of dumping and reloading.

Database maintenance brings into question concurrency and recovery. There was no attempt in this study to evaluate those functions. The multiuser tests alluded to earlier included concurrent updates as well as updates executing concurrently with queries. All systems handled such situations.

DATATRIEVE and S1032 do no journaling. With INGRES, it is optional and was never activated. Rdb has two types of journaling: after-image and before-image. The after-image journaling was never exercised; however, the before-image journaling is automatic and occasionally caused such problems as the journal file growing to 30000 blocks, exceeding user disk quota and causing the process to abort. This may be something future Rdb releases will amend, or it may be up to the user to monitor transaction lengths carefully.

## 7. GENERAL OBSERVATIONS AND COMPARISON OF FEATURES

In this final section, general observations are given, many of which are already obvious from Sects. 5 and 6. Also, features and functions not included elsewhere in this report are discussed. As stated earlier, the products will not be assigned a ranking; however, conclusions are drawn pertaining to their relative performance, ease of use, and relational functionality.

In comparison with the other three products, INGRES—with its forms, graphs, utility procedures, and interactive and embedded query language—is the most complete system. It conforms closely to the relational model, as does Rdb, but it is a more mature product with user interfaces fully developed and integrated into the system. INGRES requires more centralized control and more effort from both the INGRES system manager and the VMS manager. It is the only one of the products with a size restriction likely to cause concern—maximum record size of 2008 bytes. All the products are quite liberal in the numbers of relations, fields, records, and indexes allowed.

Indexes have a greater impact on performance of S1032 and DATATRIEVE than of INGRES and Rdb. S1032 is efficient in use and maintenance of indexes. The same good performance is seen with all indexes and not, as in DATATRIEVE and to a less extent in INGRES, only with the primary index.

Data manipulation language is probably more subject to personal preference than any other component of a database management system. S1032's language is both a positive factor and a negative one. It is very natural, meaning conversational and readable, yet not verbose. It permits extensive use of abbreviations, which reduces readability but facilitates interaction. The data manipulation language of S1032 is the most procedural of the products tested. This has the advantage of being more like familiar programming languages and affording flexibility in developing procedures. On the other hand, this is viewed as a disadvantage on the basis of the relational model and its vocabulary of set operations.

Language in INGRES and Rdb is independent of database structure. An operation is expressed one way and can be performed whether or not indexes are defined. The query optimizer determines the execution path. In DATATRIEVE and S1032, however, language is dependent on data structure, and there is no query optimizer. Especially in S1032, different syntax is required for indexes. Lack of a query optimizer means the command language must instruct the system in how to do the operation as well as tell the system what results are wanted.

In Sects. 5 and 6, there were many examples of the interactive data definition and manipulation languages. Figures 7.1, 7.2, and 7.3 show the use of these languages with a host programming language, in this case FORTRAN. DATATRIEVE is not included because it was not a part of Phase I, where the host language interface was evaluated. It has a callable interface and thus is similar to S1032.

```

.....

## declare

## integer*4 num_of_recs

## ingres mfe

## range of s is sunset
## range of n is nbiset

## retrieve (num_of_recs = count(s.shotnum where s.shotnum=n.shotnum
##           and n.nbeams=1))
## {
##   type *, ' Number of records:', num_of_recs
## }

```

Fig. 7.1. INGRES host language interface.

```

.....

.....

integer*4 num_of_recs

&Rdb& INVOKE DATABASE FILENAME "MFE"

&Rdb& GET num_of_recs = COUNT OF S IN SUNSET CROSS
&Rdb&     N IN NBISSET OVER SHOTNUM
&Rdb&     WITH N.NBEAMS=1
&Rdb& END_GET

type *, ' Number of records:', num_of_recs

```

Fig. 7.2. Rdb host language interface.

```

.....

common/S1032.COM/ num_of_recs
integer*4 num_of_recs

call DM_BEGIN      ! Initializes S1032

call DM_COMMON('nrec',%descr(num_of_recs))

call DM_CMD('open db mfe')
call DM_EXEC('set nbiset; f nb eq 1; map to sunset via an;
1             let nrec = $nrec')

type *.' Number of records:', num_of_recs

call DM_END        ! terminates S1032

```

Fig. 7.3. S1032 host language interface.

As seen in these examples, INGRES and Rdb commands are embedded in the host language program. Preprocessors, which translate and optimize the database commands, are provided for many of the VMS-supported languages. One set of commands serves the user in both interactive work and database access through a program. This is a more tightly coupled interface than that of S1032. In S1032, a set of 23 callable procedures constitutes the host language interface. In addition to learning new commands, the S1032 programmer must be concerned with buffers and static memory, especially when transferring much data between the program and the database.

Not having its own access method keeps DATATRIEVE from being a database management system, but it also gives DATATRIEVE its distinct advantage: use with RMS files. For causal use of small databases which do not require restructuring, DATATRIEVE is a reasonable choice. Data do not have to be stored multiple times. A load step is unnecessary.

Phase II experience showed DATATRIEVE to be easy to use. It was, however, the easiest to misuse, especially in the hands of a novice. In DATATRIEVE, operations can be expressed several ways. Performance statistics made it quite clear that different execution paths were taken when the command syntax was changed. Most of the 26 operations in Phase II were executed a variety of ways. What was considered the most obvious way, and which was, therefore, tried first, was in almost all cases grossly inefficient and exhibited performance out of line with the other systems. Alternate syntax was sought and found, sometimes with good results, sometimes with results which

prompted more seeking. The final recorded results were not so different from those of the other products, but there were cases where these final results were ten times better than initial attempts. DATATRIEVE documentation should contain caveats such as "use RMS utilities," "use FOR not FIND," "use primary indexes," "do RMS tuning."

Documentation is comparable for the four products: adequate but not excellent. There are omissions in the indexes of all. S1032 has too few specific examples. As stated above, DATATRIEVE documentation needs warnings and advice about which command to use.

As for vendor technical assistance, all have competent telephone support. Often answers were not immediately available, but calls were dependably returned.

Table 7.1 contains features of database systems and indicates the availability of each in the systems of this study. It is not intended to be a comprehensive collection. Some features included elsewhere in this report are not presented here. Features commonplace in current database systems are also omitted. Some items in the table have appeared earlier or are self-explanatory and will not be discussed; however, some deserve further explanation.

The table does not indicate differences in quality and implementation. In some cases, there are decided differences, such as forms and graphics which are much more extensive in INGRES than in the other systems which also have them. Another example is arrays. Arrays are not included in the relational model. They are useful in some applications and are accommodated in both DATATRIEVE and S1032. Accessing a specific element of an array is awkward in DATATRIEVE; query syntax does not include array subscripts. Array manipulation is handled with facility by S1032; there are few restrictions.

INGRES does not differentiate between missing values and zeroes in numeric fields and blanks in character fields. If missing data require special treatment, such as being excluded from statistical functions, the user must represent it in some unique way. Without a common designation for missing values, databases of less general use are created.

S1032 is given a check for report writer even though it has no separate component for that specific purpose. Its procedural language and flexible formatting provide report generation capability.

An effort was made in defining the two databases for this evaluation to include all common VMS datatypes. One exception was packed decimal, which only DATATRIEVE and S1032 support. A new datatype is introduced by Rdb: segmented string. In early development of Rdb, this was referred to as a "blob" because of its size (maximum of 65K bytes) and its unformatted structure. This datatype allows the storage of large amounts of text, such as abstracts or source code, or long strings of binary data. Most Rdb operations are not supported for segmented strings; however, they can be accessed and processed by programs.

In conclusion, DATATRIEVE, INGRES, Rdb, and S1032 are not equal in their performance, functionality, and usability. They each have strengths and weaknesses, advantages and disadvan-

Table 7.1. Features of database systems

Feature	DTR	INGRES	Rdb	S1032
ARRAYS	✓	-	-	✓
FORMS	-	✓	-	v4
GRAPHICS	✓	✓	-	-
ON-LINE HELP	✓	✓	✓	✓
COMPILED PROCS	-	-	-	✓
QUERY OPTIMIZING	-	✓	✓	-
MISSING VALUE	✓	-	✓	✓
VALIDITY CHECKING	✓	✓	✓	✓
RMS INTERFACE	✓	✓	-	✓
INIT FILE	✓	✓	✓	✓
SESSION LOG	✓	✓	✓	-
REPORT WRITER	✓	✓	-	✓
VARIABLE LEN REC	✓	✓	✓	-
STATISTICAL FUNCTIONS	✓	✓	✓	✓
RTL INTERFACE	✓	✓	-	✓
UNRESTRICTED JOIN	-	✓	✓	-
VIEWS	✓	✓	✓	-
B-TREE INDEXING	-	v3	✓	✓
PACKED DECIMAL DATA	✓	-	-	✓
JOURNALING	-	✓	✓	-
SECURITY	✓	✓	✓	✓
DECnet SUPPORT	✓	✓	✓	✓

tages. None exhibits anomalous behavior which would *a priori* eliminate it from consideration. The answer to the question "which database system to use" is the question, "what is the application?". By making a rational extrapolation from the results given in this report to the parameters of the reader's problem, it may be possible to select a best product or eliminate some of the contenders. In the uncharted waters of software selection, perhaps this approach will help maintain orientation even if it does not prescribe an exact course.



## INTERNAL DISTRIBUTION

1. R. K. Abercrombie
2. E. F. Abercrombie
3. B. L. Alspaugh
4. D. H. Alspaugh
5. B. T. Anderson
6. L. L. Anthony
7. Z. Batte
8. L. R. Baylor
9. J. D. Bell
10. R. R. Bentz
11. M. B. Biddix
12. K. L. Boylan
13. R. W. Browell
14. S. L. Bunch
15. R. D. Burris
16. D. N. Clark
17. A. L. Coffie
18. T. B. Cook
19. J. B. Cordts
20. B. B. Corey
21. J. G. Craven
22. C. M. Davenport
23. J. Defenderfer
24. J. H. Dixon
25. R. A. Dory
26. J. R. Drake
27. L. Duncan
28. T. H. Dumigan
29. P. H. Edmonds
30. J. E. Edwards
31. W. B. Ewbank
32. R. D. Foskett
33. R. M. French
34. E. L. Frome
35. T. L. Futrell
36. D. C. Giles
37. R. O. Green
38. V. P. Gupta
39. F. D. Hammerling
40. C. E. Hammons
41. M. B. Heath
42. T. Heath
43. H. H. Hogue
44. B. M. Horwedel
45. J. E. Horwedel
46. J. V. Hughes
47. D. F. Hunt
48. R. C. Isler
49. J. M. Jansen, Jr.
50. C. Jelener
51. G. W. Joe
52. C. K. Johnson
53. V. B. Johnson
54. J. G. Jones
55. S. R. Jordan
- 56-75. K. L. Kamman
76. H. E. Ketterer
77. P. W. King
78. E. S. Krebs
79. L. R. Layman
80. A. M. Lokey
81. W. W. Manges
82. J. B. Mankin
83. D. M. McCloud
84. J. L. McNeary
85. S. H. Merriman
86. K. C. Miller
87. D. L. Million
88. D. L. Morgan
89. J. K. Mumro
90. R. E. Neal
91. C. Ooten
92. D. R. Overbey
93. C. M. Pflieger
94. W. R. Pickett
95. C. E. Price
96. M. S. Pung
97. A. B. Ralph
98. W. R. Reagan
99. B. H. Robbins
100. T. E. Rowe
101. M. J. Saltmarsh
102. H. S. Schwartz
103. R. J. Sharp
104. R. L. Shipp
105. W. D. Sims
106. K. W. Stanton
107. R. W. Taylor
108. D. H. Thompson
109. H. G. Travis
110. T. C. Tucker
111. R. J. Verastegui
112. B. D. Vickers
113. P. W. Wallace
114. J. E. White
115. W. R. Wing
116. J. W. Wooten
117. K. G. Young
- 118-119. Laboratory Records Department
120. Laboratory Records, ORNL-RC
121. Document Reference Section
122. Central Research Library
123. Fusion Energy Division Library
- 124-125. Fusion Energy Division  
Publications Office
126. ORNL Patent Office

## EXTERNAL DISTRIBUTION

127. A. A. Brooks, 100 W. Wiltshire Dr., Oak Ridge, TN 37830
128. T. W. Fredian, Bldg. NW17, Rm. 262, Albany Street, Cambridge, MA 02139
129. A. Macmahon, Fusion Research Center, University of Texas at Austin, RLM 11.222, Austin, TX 78712
130. L. Mann, Controlled Thermonuclear Research Division, Los Alamos National Laboratory, MS-F647, Los Alamos, NM 87545
131. P. Pearson, Lawrence Livermore National Laboratory, P.O. Box 808 (L-560), Livermore, CA 94550
132. W. Pfeiffer, GA Technologies, TO-503B, P.O. Box 85600, San Diego, CA 92138
133. D. Hitchcock, Fusion Theory and Computer Services Branch, Applied Plasma Physics, Office of Fusion Energy, Office of Energy Research, ER-55, GTN, U.S. Department of Energy, Washington, DC 20545
134. P. H. Diamond, Institute for Fusion Studies, University of Texas at Austin, Austin, TX 78712
135. Office of the Assistant Manager for Energy Research and Development, Department of Energy, Oak Ridge Operations Office, P.O. Box E, Oak Ridge, TN 37831
136. J. D. Callen, Department of Nuclear Engineering, University of Wisconsin, Madison, WI 53706
137. R. W. Conn, Department of Chemical, Nuclear, and Thermal Engineering, University of California, Los Angeles, CA 90024
138. S. O. Dean, Director, Fusion Energy Development, Science Applications International Corp., Gaithersburg, MD 20760
139. H. K. Forsen, Bechtel Group, Inc., Research Engineering, P.O. Box 3965, San Francisco, CA 94205
140. J. R. Gilleland, GA Technologies, Inc., Fusion and Advanced Technology, P.O. Box 85608, San Diego, CA 92138
141. R. W. Gould, Department of Applied Physics, California Institute of Technology, Pasadena, CA 91125
142. R. W. Gross, Plasma Research Library, Columbia University, New York, NY 10027
143. D. M. Meade, Princeton Plasma Physics Laboratory, P.O. Box 451, Princeton, NJ 08544
144. W. M. Stacey, School of Mechanical Engineering, Georgia Institute of Technology, Atlanta, GA 30332
145. D. Steiner, Rensselaer Polytechnic Institute, Troy, NY 12181
146. R. Varma, Physical Research Laboratory, Navrangpura, Ahmedabad 380009, India
147. Bibliothek, Max-Planck Institut fur Plasmaphysik, D-8046 Garching bei Munchen, Federal Republic of Germany
148. Bibliothek, Institut fur Plasmaphysik, KFA, Postfach 1913, D-5170 Julich, Federal Republic of Germany
149. Bibliotheque, Centre des Recherches en Physique des Plasmas, 21 Avenue des Bains, 1007 Lausanne, Switzerland
150. Bibliotheque, Service du Confinement des Plasmas, CEA, B.P. No. 6, 92 Fontenay-aux-Roses (Seine), France
151. Documentation S. I. G. N., Department de la Physique du Plasma et de la Fusion Controlee, Centre d'Etudes Nucleaires, B.P. 85, Centre du Tri, 38081 Cedex, Grenoble, France
152. Library, Culham Laboratory, UKAEA, Abingdon, Oxfordshire, OX14 3DB, England
153. Library, FOM-Instituut voor Plasma-Fysica, Rijnhuizen, Edisonbaan 14, 3439 MN Nieuwegein, The Netherlands
154. Library, Institute of Plasma Physics, Nagoya University, Nagoya, Japan
155. Library, International Centre for Theoretical Physics, Trieste, Italy
156. Library, Laboratorio Gas Ionizzati, CP 56, I-00044 Frascati (Roma), Italy
157. Library, Plasma Physics Laboratory, Kyoto University, Gokasho, Uji, Kyoto, Japan
158. Plasma Research Laboratory, Australian National University, P.O. Box 4, Canberra, A.C.T. 2000, Australia
159. Thermonuclear Library, Japan Atomic Energy Research Institute, Tokai, Naka, Ibaraki, Japan

160. G. A. Eliseev, I. V. Kurchatov Institute of Atomic Energy, P.O. Box 3402, 123182 Moscow, U.S.S.R.
161. V. A. Glukhikh, Scientific-Research Institute of Electro-Physical Apparatus, 188631 Leningrad, U.S.S.R.
162. I. Shpigel, Institute of General Physics, U.S.S.R. Academy of Sciences, Ulitsa Vavilova 38, Moscow, U.S.S.R.
163. D. D. Ryutov, Institute of Nuclear Physics, Siberian Branch of the Academy of Sciences of the U.S.S.R., Sovetskaya St. 5, 630090 Novosibirsk, U.S.S.R.
164. V. T. Tolok, Kharkov Physical-Technical Institute, Academical St. 1, 310108 Kharkov, U.S.S.R.
165. Library, Institute of Physics, Academia Sinica, Beijing, Peoples Republic of China
166. J. F. Clarke, Associate Director for Fusion Energy, Office of Fusion Energy, ER-50, Germantown, U.S. Department of Energy, Washington, DC 20545
167. D. B. Nelson, Division of Applied Plasma Physics, Office of Fusion Energy, Office of Energy Research, ER-54, Germantown, U.S. Department of Energy, Washington, DC 20545
168. N. A. Davies, Tokamak Systems Branch, Office of Fusion Energy, Office of Energy Research, ER-55, Germantown, U.S. Department of Energy, Washington, DC 20545
169. E. Oktay, Tokamak Systems Branch, Office of Fusion Energy, Office of Energy Research, ER-55, Germantown, U.S. Department of Energy, Washington, DC 20545
170. Theory Department Read File, c/o D. W. Ross, University of Texas, Institute for Fusion Studies, Austin, TX 78712
171. Theory Department Read File, c/o R. C. Davidson, Director, Plasma Fusion Center, NW 16-202, Massachusetts Institute of Technology, Cambridge, MA 02139
172. Theory Department Read File, c/o F. W. Perkins, Princeton Plasma Physics Laboratory, P.O. Box 451, Princeton, NJ 08544
173. Theory Department Read File, c/o L. Kovrizhnykh, Institute of General Physics, Academy of Sciences of the U.S.S.R., Ulitsa Vavilova 38, Moscow, U.S.S.R.
174. Theory Department Read File, c/o B. B. Kadomtsev, I. V. Kurchatov Institute of Atomic Energy, P.O. Box 3402, 123182 Moscow, U.S.S.R.
175. Theory Department Read File, c/o T. Kamimura, Institute of Plasma Physics, Nagoya University, Nagoya, Japan
176. Theory Department Read File, c/o C. Mercier, Euratom-CEA, Service des Recherches sur la Fusion Contrôlée, Fontenay-aux-Roses (Seine), France
177. Theory Department Read File, c/o T. E. Stringer, JET Joint Undertaking, Culham Laboratory, Abingdon, Oxfordshire OX14 3DB, England
178. Theory Department Read File, c/o K. Roberts, Culham Laboratory, Abingdon, Oxfordshire OX14 3DB, England
179. Theory Department Read File, c/o D. Biskamp, Max-Planck-Institut für Plasmaphysik, D-8046 Garching bei München, Federal Republic of Germany
180. Theory Department Read File, c/o T. Takeda, Japan Atomic Energy Research Institute, Tokai, Naka, Ibaraki, Japan
181. Theory Department Read File, c/o C. S. Liu, GA Technologies, Inc., P.O. Box 81608, San Diego, CA 92138
182. Theory Department Read File, c/o L. D. Pearlstein, Lawrence Livermore National Laboratory, P.O. Box 808, Livermore, CA 94550
183. Theory Department Read File, c/o R. Gerwin, CTR Division, Los Alamos National Laboratory, P.O. Box 1663, Los Alamos, NM 87545
184. R. E. Mickens, Department of Physics, Atlanta University, Atlanta, GA 30314
185. C. De Palo, Library, Associazione EURATOM-ENEA sulla Fusione, CP 65, I-00044 Frascati (Roma), Italy
- 186-341. Given distribution as shown in TIC-4500 Magnetic Fusion Energy (Category Distribution UC-20 g: Theoretical Plasma Physics)