NCE & TECHNOLOGY:

MODELING AND MEASUREMENT TECHNIQUES FOR EVALUATION OF DESIGN ALTERNATIVES IN THE IMPLEMENTATION OF DATABASE MANAGEMENT SOFTWARE

DEPARTMENT OF COMMERCE • NATIONAL BUREAU OF STANDARDS •

NBS Special Publication 500-49
U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards

NATIONAL BUREAU OF STANDARDS

The National Bureau of Standards[1] was established by an act of Congress March 3, 1901. The Bureau's overall goal is to strengthen and advance the Nation's science and technology and facilitate their effective application for public benefit. To this end, the Bureau conducts research and provides: (1) a basis for the Nation's physical measurement system, (2) scientific and technological services for industry and government, (3) a technical basis for equity in trade, and (4) technical services to promote public safety. The Bureau's technical work is performed by the National Measurement Laboratory, the National Engineering Laboratory, and the Institute for Computer Sciences and Technology.

THE NATIONAL MEASUREMENT LABORATORY provides the national system of physical and chemical and materials measurement; coordinates the system with measurement systems of other nations and furnishes essential services leading to accurate and uniform physical and chemical measurement throughout the Nation's scientific community, industry, and commerce; conducts materials research leading to improved methods of measurement, standards, and data on the properties of materials needed by industry, commerce, educational institutions, and Government; provides advisory and research services to other Government Agencies; develops, produces, and distributes Standard Reference Materials; and provides calibration services. The Laboratory consists of the following centers:

Absolute Physical Quantities[2] — Radiation Research — Thermodynamics and Molecular Science — Analytical Chemistry — Materials Science.

THE NATIONAL ENGINEERING LABORATORY provides technology and technical services to users in the public and private sectors to address national needs and to solve national problems in the public interest; conducts research in engineering and applied science in support of objectives in these efforts; builds and maintains competence in the necessary disciplines required to carry out this research and technical service; develops engineering data and measurement capabilities; provides engineering measurement traceability services; develops test methods and proposes engineering standards and code changes; develops and proposes new engineering practices; and develops and improves mechanisms to transfer results of its research to the utlimate user. The Laboratory consists of the following centers:

Applied Mathematics — Electronics and Electrical Engineering[2] — Mechanical Engineering and Process Technology[2] — Building Technology — Fire Research — Consumer Product Technology — Field Methods.

THE INSTITUTE FOR COMPUTER SCIENCES AND TECHNOLOGY conducts research and provides scientific and technical services to aid Federal Agencies in the selection, acquisition, application, and use of computer technology to improve effectiveness and economy in Government operations in accordance with Public Law 89-306 (40 U.S.C. 759), relevant Executive Orders, and other directives; carries out this mission by managing the Federal Information Processing Standards Program, developing Federal ADP standards guidelines, and managing Federal participation in ADP voluntary standardization activities; provides scientific and technological advisory services and assistance to Federal Agencies; and provides the technical foundation for computer-related policies of the Federal Government. The Institute consists of the following divisions:

Systems and Software — Computer Systems Engineering — Information Technology.

[1]Headquarters and Laboratories at Gaithersburg, Maryland, unless otherwise noted; mailing address Washington,D.C. 20234.
[2]Some divisions within the center are located at Boulder, Colorado, 80303.

The National Bureau of Standards was reorganized, effective April 9, 1978.

COMPUTER SCIENCE & TECHNOLOGY:

Modeling and Measurement Techniques for Evaluation of Design Alternatives in the Implementation of Database Management Software

Donald R. Deutsch

Center for Programming Science and Technology
Institute for Computer Sciences and Technology
National Bureau of Standards
Washington, D.C. 20234

U.S. DEPARTMENT OF COMMERCE, Juanita M. Kreps, Secretary

Jordan J. Baruch, Assistant Secretary for Science and Technology

NATIONAL BUREAU OF STANDARDS, Ernest Ambler, Director

Issued July 1979

Reports on Computer Science and Technology

The National Bureau of Standards has a special responsibility within the Federal Government for computer science and technology activities. The programs of the NBS Institute for Computer Sciences and Technology are designed to provide ADP standards, guidelines, and technical advisory services to improve the effectiveness of computer utilization in the Federal sector, and to perform appropriate research and development efforts as foundation for such activities and programs. This publication series will report these NBS efforts to the Federal computer community as well as to interested specialists in the academic and private sectors. Those wishing to receive notices of publications in the series should complete and return the form at the end of this publication.

National Bureau of Standards Special Publication 500-49
Nat. Bur. Stand. (U.S.), Spec. Publ. 500-49, 244 pages (July 1979)
CODEN: XNBSAV

Library of Congress Catalog Card Number: 79-600088

U.S. GOVERNMENT PRINTING OFFICE
WASHINGTON: 1979

For sale by the Superintendent of Documents, U.S. Government Printing Office, Washington, D.C. 20402
Stock No. 003-003-02088-5 Price $5.50
(Add 25 percent additional for other than U.S. mailing).

PREFACE

This report is comprised of a dissertation submitted to the Faculty of the Graduate School of the University of Maryland in partial fulfillment of the requirements for the degree of Doctor of Business Administration. The degree was conferred in December 1978 with a dual major in Management Science and Information Systems Management, and with minors in Accounting, Organization Theory and Management, and Marketing. Members of the doctoral dissertation committee are listed below.

Co-Chairmen:

Dr. Edgar H. Sibley
Professor and Acting Chairman
Department of Information Systems Management
University of Maryland

Dr. Saul I. Gass
Professor and Chairman
Management Science
College of Business and Management
University of Maryland

Members:

Dr. W. Terry Hardgrave
Assistant Professor
Department of Information Systems Management
University of Maryland

Dr. Dennis W. Fife
Chief, Applications Systems Division
Center for Programming Science and Technology
National Bureau of Standards

Dr. Rudolph P. Lamone
Professor and Dean
College of Business and Management
University of Maryland

The research described herein was performed, in part, in the author's capacity as a Computer Scientist at the National Bureau of Standards' Institute for Computer Sciences and Technology and as an Instructor in the University of Maryland's Department of Information Systems Management. The project and related research received support in the form of funding, personnel or other resources from both the National Bureau of Standards and the University of Maryland.

ACKNOWLEDGMENTS

I am grateful to many individuals and organizations for assisting me in this research. At the University, I received scholarly tutelage and direction. At the National Bureau of Standards, I was given the environment and support necessary for my research. And at home, I found encouragement and understanding. With sincerest apologies to those not mentioned, I express my appreciation to the following people:

* to Ed Sibley, for introducing me to database management and guiding my academic and professional progress;
* to Dennis Fife, for giving me guidance as well as the time and resources that I needed for this research;
* to Terry Hardgrave, for his friendship and the numerous hours, countless cups of coffee, and more than a few beers he invested in this research;
* to the other members of my dissertation committee, Saul Gass and Rudy Lamone, and faculty members of the Department of Information Systems Management, for their help throughout this project;
* to those who worked on the set processor and related software at NBS over the years, including: Ray Somers, Erica Jen, Chuck Sheppard, Tony Marriott, and Margaret Moyer;
* to my colleagues and friends at NBS for tolerating and helping me in all phases of this work, especially Linda Ross and Barbara Peterson who typed this document;
* and most importantly to my wife, Martha, whose patience, understanding, encouragement and love provided me with the strength and motivation to complete this research.

TABLE OF CONTENTS

TABLE OF FIGURES

MODELING AND MEASUREMENT TECHNIQUES FOR THE EVALUATION OF DESIGN ALTERNATIVES IN THE IMPLEMENTATION OF DATABASE MANAGEMENT SOFTWARE

Donald R. Deutsch

The substantial costs associated with building complex hardware/software systems make the traditional development approach of implementation followed by several iterations for modification and enhancement unacceptable for building modern database management systems. Mechanisms for determining gross feasibility prior to the commitment of resources for major software development efforts are required. An integrated approach combining the development of a limited but well-structured DBMS prototype with the use of high-level measurement and predictive modeling techniques for evaluating design alternatives in the implementation of database management software is proposed as an alternative to the traditional development - enhancement spiral.

Using a prototype for a set-theoretic implementation of a database management system with a relational user interface as an object, this research demonstrates that proposed DBMS designs can be evaluated through the use of performance prediction models based on prototype implementations and associated measurement systems.

Key words: Analytic models; database management; model validation; performance evaluation; performance measurement; predictive modeling; set-processing; simulation; software design.

1. PROBLEM STATEMENT

1.1 Background and Motivation

Generalized database management systems (DBMS) are being used by increasing numbers of organizations in both the public and private sectors [IDC78]. Despite this growing acceptance, implementations of large scale systems using DBMS software are still limited by both technological and economic constraints. Very large database applications may be too costly and/or may exceed current technological limits in many ways, including the following.

* Secondary storage requirements can be excessive even with decreasing costs and increasing capacities for direct access media; database storage requirements can exceed hardware capabilities or can become too expensive relative to application benefits.

* Response times can exceed those acceptable for the application; conversely, adequate response times may require processor capabilities that exceed current technological limits.

* The complexity and power of DBMS software may monopolize existing machine resources; the acquisition of additional computer capacity may be economically unjustifiable.

Limitations inherent in DBMS products currently available are responsible, at least in part, for intensive study and research efforts addressing database management system design concepts in industrial, university and government laboratories [ASTR76, ZLOO75, STON76, LIN76, CHEN76, OZKA77, HARD76a-b]. This quest can be expected to continue until database management systems reach a level of development where they can be economically applied to all problems that would benefit from their use.

Recent developments in database concepts and technology promise DBMS designs that are radically different from those in the marketplace today. In particular, the relational data model proposed by Codd [CODD70, CODD71, CODD72a-b] and the extensions to set theory presented by Childs [CHIL68a-b, CHIL77] and Hardgrave [HARD76a-b] have received a great deal of attention from the academic and research communities. While these concepts promise a substantial improvement over existing systems, the performance potential of new DBMS designs incorporating these ideas is not known.

Traditionally, database management software has been developed in an ad-hoc manner; design ideas have been tested by implementation. The resulting DBMS software has then been iteratively tuned and often rewritten to achieve acceptable levels of performance. The possibility that underlying design concepts may have inherent performance limitations has not been considered prior to committing resources for software development.

The use of predictive models to evaluate and guide DBMS design decisions is not common. When they are applied, modeling techniques are generally used to evaluate performance characteristics of system components. Few attempts have been made to evaluate entire DBMS design frameworks prior to their implementation; prominent research prototypes for relational DBMS's were developed without the benefit of mathematical evaluation of design alternatives [STON77,DEJO79].

The research described herein was motivated by the conviction that an alternative to the traditional implementation - enhancement approach to DBMS development was needed; the increasing costs associated with building complex hardware/software systems such as DBMS preclude continued use of the ad-hoc "build it and see if it flies" methods of the past. Just as the airplane builder must be reasonably certain that a new plane design will fly, the DBMS implementor should have some prior assurance that the system will perform satisfactorily within pertinent technological and economic constraints.

1.2 Research Objectives

A product of a continuing research project concerned with developing a methodology for evaluating proposed DBMS designs, this research report describes and demonstrates a database management system design evaluation and development approach that combines, in an integrated effort, the following components.

* Development of a limited prototype DBMS.

* Use of a flexible, high level measurement facility.

* Estimation of DBMS performance potential using a predictive model.

This integrated approach is proposed as a preferable alternative to current practice.

A database management system design incorporating several state-of-the-art concepts is used as an object in this research. While addressing the questions of efficiency and feasibility for DBMS implementations based on these ideas, the purpose of the research is not to prove the efficacy of the design concepts. The measure of success is the ability of the proposed integrated evaluation approach to provide insight into the performance potential for DBMS designs. These research objectives are stated more formally in the thesis statement and propositions below.

1.3 Thesis Statement

The major proposition addressed by this research is embodied in the following thesis statement:

> Proposed database management system designs can be evaluated best through the integrated use of a limited prototype implementation for the DBMS design, a flexible measurement facility, and a predictive model based on the DBMS prototype.

In the process of considering this thesis, the following minor or supporting propositions were formulated and addressed by the research.

1.3.1 Proposition-1. The model can be developed at a high level; that is, DBMS programs rather than operating system events can be considered.

1.3.2 Proposition-2. A simple and flexible high-level measurement system can be developed for monitoring prototype DBMS performance.

1.3.3 Proposition-3. A methodology using measured prototype performance data for understanding DBMS operation, for deriving parameter values and calibrating the model, and for validating model predictions can be developed.

1.4 Overview of Research Report

This report describes a research effort, that was carried out over a period of several years, to develop and demonstrate a methodology for evaluating proposed DBMS designs. The next chapter reviews pertinent literature and related research. Chapter 3 presents an overview of a performance prediction system with three components: a prototype implementation for a positional set processor DBMS, a measurement and analysis system, and a performance prediction model. Chapters 4 and 5 describe the DBMS prototype and measurement facility. Model concepts and parameters are discussed in chapter 6; chapter 7 summarizes the mathematical relationships imbedded in the performance prediction model. Chapter 8 considers the model evaluation problem and reviews the status of ongoing verification, validation and problem analysis activities. Chapters 9 and 10 present research results and chart future research directions. Finally, Appendices containing scripts demonstrating the set processor DBMS prototype and performance modelers follow an alphabetical Bibliography and list of references.

2. PREVIOUS AND RELATED RESEARCH

2.1 Computer Performance Evaluation

Computer hardware/software facilities are complex and interesting systems. Their performance is not easy to understand or predict. A number of techniques are used for monitoring and evaluating computer system performance [LUCA71, SVOB76], including:

* analysis of times for hardware cycles, weighted mixes of instructions, or typical "kernel" programs;
* execution of benchmarks and synthetic programs representing projected system load;
* performance monitoring;
* analytical modeling; and
* simulation modeling.

Time analyses emphasize raw hardware speed, and do not adequately address systems software characteristics for predicting performance. Benchmarks and synthetic programs are useful primarily for selecting among alternative existing computer systems based on a projected system load; they are not adequate for predicting performance of non-existent hardware or software. Monitoring tools are used primarily to collect data on the performance of existing hardware/ software systems for locating bottlenecks, improving performance and guiding administrative policies. Only analytic and simulation modeling have the power for predicting performance of proposed hardware and software designs as well as for evaluating existing computer system operations.

2.2 Models of Computer Systems

While mathematical models of computer systems were useful for evaluating and predicting the performance of (single user) serial processing machines [KLEI66], computer system modeling increased dramatically with the advent of multiprogramming. SCHA67 and BASK71 are compendia of both analytic and simulation models for multi-user computer systems; bibliographies appear in ANDA72 and SVOB76. Analytic and simulation models are considered separately below.

2.2.1 Analytic Models. Analytic models for computer system performance evaluation are mathematical representations of computing systems that are solved using the tools of mathematics [SMIT66]. Analytic models often address performance characteristics for particular computer system components or functions, such as disk I/O [FIFE65] and job scheduling [LUCA71, BASK71]. Particularly common are applications of queueing theory to time-sharing through-put analysis [KLEI64, CHAN66, COFF67 & 68, BUZE73, BOYS75, LIPS77]. Analytic models are generally unique to each application. GRAH78 is a monograph that surveys the current state-of-the-art for queueing network models of computer system performance. High costs for analytic models and their limited abilities for representing complex interactions have motivated the development of computer system simulation models.

2.2.2 Simulation Models. The most flexible and potentially powerful technique for predicting computer system performance is simulation [LUCA71]. Computer systems are usually modeled in terms of system states and state transitions or events occurring at discrete time intervals [FERR78]. Simulators have been frequently used to evaluate the design of particular hardware/software systems [YOUC64, KATZ67]; some have been successfully generalized to represent computer systems other than the original model object [NIEL67].

Powerful computer system simulators claim to be applicable to broad classes of hardware and systems software that are parametrically described in extensive factor libraries [HUES67, IHRE67, SEAM69]. Two commercial computer simulation products, SCERT (COMRESS Div. of Comten), and CASE (Testdata Systems), are described and compared in FERR78. ECSS, a language for developing computer system simulations, is described by KOSY73.

Trace driven modeling combines measurement and simulation in a manner that overcomes some of the difficulties associated with validating pure simulation models that are based on distributions and random variables. Trace driven models use measured event sequences instead of probability distributions to describe resource requirements [SHER76]. This special type of computer system simulation has been applied to both partial and complete hardware/software systems [SHER73, CHEG69, NOE72, SHER72].

Computer system simulation is addressed by a growing body of literature [FERR78, HIGH73-76]. A tutorial description of the construction of a basic simulation model for a multiprogrammed computer system and an annotated bibliography appear in MACD70.

2.3 Models of Database Management Systems

Models of database management systems are comparatively rare and recent research tools. Focusing on analyzing the behavior of existing systems and developing simple probabilistic representations for specific functional components, DBMS models consider the impact of changes on system performance [BLAS75].

Senko and his colleagues were some of the first to recognize the need for database management system models. The landmark FOREM research produced a comprehensive collection of file organization evaluation models [SENK67, SENK70] and a file design handbook containing guidelines for selecting access methods and determining secondary storage utilization strategies [SENK69]. The follow-on Phase-II model was a special purpose simulation for predicting execution time of computer systems dedicated to database management applications [OWEN71].

<u>2.3.1 Models of DBMS components and functions.</u> Database system components and functions have been the objects of both analytical and simulation studies. FOREM was primarily a collection of deterministic tools for analyzing storage structures, although simulation was used for deriving file design guidelines. Cardenas [CARD73] and Yao [YAO74, YAO77] applied analytic techniques to the evaluation and selection of file organizations.

An objective function for minimizing secondary index costs is presented in ANDD77. Secondary indices are analyzed deterministically in CARD75. Others have addressed secondary index selection and design problems using various techniques and assumptions [LUM71, SCHK75, SHNE74].

The assignment of data collections to alternative storage media, devices and nodes in a network has been the object of modeling research [LUM75, BUZE74, CASE72]. A survey of physical database design techniques including modeling approaches appears in SCHK78.

The impact on database management system performance of virtual storage management strategies is the focus of four recent studies [LANG77, BRIC77, SHER76, FERN78]. In their work, Sherman and Brice used trace driven modeling techniques for studying DBMS virtual memory system interactions.

Modeling techniques have been used to investigate database reorganization strategies. A performance model using queueing analysis for studying database reorganization performed concurrently with usage is presented by Sockut [SOCK78].

2.3.2 DBMS modeling languages. Most database management simulations have been implemented using high-level programming languages such as FORTRAN. Two DBMS models developed using generalized tools appear in the literature. Nakamura et. al. describe a simulation model written using a computer system simulation package [NAKA75]. A model for evaluating the capability of a hardware vendor provided DBMS to handle a particular large, real-time facility assignment and inventory system was developed using the GPSS simulation language [GRIF75].

2.3.3 File organization models. Severance has applied both analytical and simulation techniques to the study of data and storage structures. Addressing various aspects of the file organization evaluation problem, this research is recorded in numerous working papers and articles [SEVE72, SEVE74b, SEVE75, SEVE76b]. Products of his modeling efforts comprise a comprehensive collection of normative tools for aiding the database designer. Specific problems addressed include: record and file segmentation and blocking factors [MARC76, EISN76, SEVE76a], access path selection [SEVE77], and search mechanism evaluation [SEVE74a]. MARC78 synthesizes much of the database design research appearing in the other references in this paragraph into a generalized model of secondary memory management.

The evaluation of storage structures in terms of performance for a given retrieval workload is addressed by SCHE76. Scheuermann presents a development methodology and a preliminary implementation of a simulation model for guiding the selection of storage structures. His goal was to develop a systematic methodology for designing a simulation model to aid database administrators in selecting file organizations.

2.3.4 Generalized DBMS models. Two researchers have attempted to develop generalized models applicable to a broad range of data base management systems. Reiter has a generalized simulation modeling framework that is tailored to specific object DBMS's by the addition of "plug-in" user written FORTRAN models. In what he terms an inductive approach, Reiter's modeling process has three stages:

* Translate - a task description specified by the user at a high conceptual level is translated into an intermediate level system representation.

* Synthesize - a representation modeler maps logical data elements into blocks of secondary storage.

* Execute - system dynamics are simulated by an execution modeler for a multiprogrammed operating system.

Called DIMUI in recent publications, Reiter's model and its application are described in REIEa-b, REIE77a and SHNE78. A discussion of the role of simulation in evaluating database management systems using DIMUI for illustrative experiments appears in [REIE77b].

DeLutis's Information Processing System Simulator (IPSS) provides facilities for modeling DBMS software and buffer management. DBMS application programs, scheduling and resource allocation functions, and data manipulation operations must be procedurally specified. Containing both PL/1 and FORTRAN code, the IPSS methodology views an implementation processing system as being a hierarchy of three types of functions:

* a database,

* services for accessing the database, and

* support facilities.

The major emphasis of the IPSS research is on the application of macro-analysis; that is, its focus is on measuring system performance from the user's perspective [DELU77]. Reiter's model, IPSS and the ECSS computer system simulation language are compared and evaluated in FEDS78.

2.4 Research Milieu

The literature reviewed above provides the setting for the research described in this document. The integrated DBMS design evaluation approach that is described herein utilizes performance monitoring and both analytic and simulation modeling techniques. Unlike most computer system modeling efforts, this research has as its objective the determination of gross performance potential rather than system tuning. This research differs also in that it focuses on DBMS software, with the operating system and hardware seen as comprising the processing environment.

This research is most closely related to the few past database management system modeling endeavors. The objective of comprehensively representing DBMS performance is similar to those described in REIE76b and DELU77. However, neither Reiter nor DeLutis can easily or directly model the set representation constructs used in the DBMS design that is the object of this modeling effort; indeed, no known previously developed model could be applied to this problem.

Like the trace driven database management system performance evaluations carried out by Sherman and Brice [SHER76, BRIC77], this research relies heavily on the use of measured performance data and event traces. While the current model is not strictly trace driven, the sequence of DBMS functions required to answer a query is determined. Derived in a manner similar to the actual DBMS, this functional sequence can be viewed as a pseudo-trace. The high-level measurement and modeling employed for this project differs from the operating system service request level used by Sherman and Brice. Their respective levels of detail reflect the different purposes of the two studies; while Sherman and Brice address specific DBMS operating system interactions, the research described herein is concerned with predicting gross performance potential.

One difference between this and all previous research efforts is the emphasis on validation; from the outset, the strongest form of predictive validity was the objective of this effort. Predictive validation was considered in only a few previous efforts [GRIF75, SHER76, BRIC77]. Previous general purpose modeling efforts have attempted only much weaker forms of validation based on relative changes and the philosophy of rationalism described in Chapter 8.

The research described in this document addresses the evaluation of proposed database management system designs. This contrasts with the database design problems addressed by Senko, Severance, Yao and others. Some of the normative results from other research endeavors could be used in extensions of the current research; the model produced for this study could be enhanced, for instance, by including storage structure optimization modules to predict the best possible performance for a given design strategy.

3. PERFORMANCE PREDICTION SYSTEM OVERVIEW

3.1 Introduction

The motivation for this research is the belief that the substantial costs associated with building complex hardware/software systems preclude the development of new database management systems using traditional techniques. Mechanisms for determining gross feasibility of proposed DBMS designs are needed; development should not begin on a DBMS design concept without prior assurance that an implementation can be made to satisfy user requirements within reasonable time and cost constraints. The traditional software development approach of implementation followed by several iterations for modification and enhancement to increase performance and efficiency is not suitable for building modern database management systems. An alternative to this development-enhancement spiral is presented by Hardgrave and Sibley [HARD75d]. They propose an approach combining the development of a limited prototype with the use of predictive modeling for understanding critical design factors and evaluating the feasibility of new designs.

This chapter presents an overview of a DBMS development approach that utilizes and extends the ideas proposed by Hardgrave and Sibley. The next section describes the use of prototype development, measurement, and modeling in an integrated effort to predict performance and guide future implementation decisions. Following a discussion of the performance prediction system components, an overview of a specific application of the proposed approach is given. The latter section, describing the Set Processor Performance Prediction System, provides the framework for the three subsequent chapters.

3.2 Integrated Development-Measurement-Modeling

Figure 3.1 is a graphical representation of an integrated prototype development, measurement and modeling approach for predicting performance and guiding development of a proposed DBMS design. For a given DBMS design concept, a performance prediction system is used to evaluate the efficacy of the proposed design and to guide full-scale DBMS product development decisions. The performance prediction system is made up of three integrated parts.

INTEGRATED DEVELOPMENT-MEASUREMENT-MODELING APPROACH

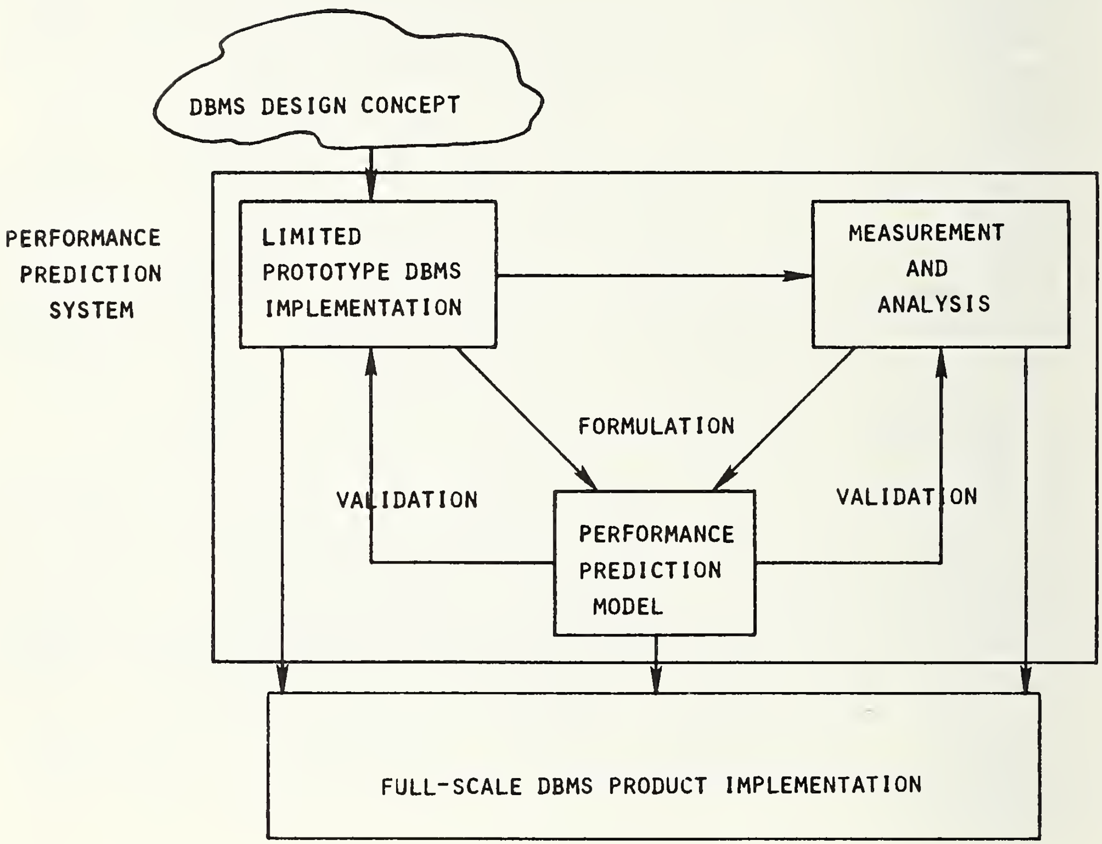


FIGURE 3.1

* A limited prototype DBMS implementation.

* A high-level measurement and analysis system.

* A performance prediction model.

When the prototype implementation is exercised, its performance is monitored and recorded by a measurement and analysis system. The prototype implementation and data generated by the measurement system about its performance provide the insight and understanding necessary for developing a performance prediction model. The model can be used for evaluating the potential of implementations using the design concepts embedded in the prototype. The prototype DBMS and measurement system also provide a mechanism for validating the model. It is then possible to perturb model parameters beyond the range of prototype DBMS capabilities to determine if and how full-scale implementation should proceed. Characteristics of the three performance prediction system components are considered in the following paragraphs.

<u>3.2.1 Prototype DBMS implementation.</u> The heart of a DBMS Performance Prediction System is a limited prototype implementation for the proposed design concept. Development of a DBMS prototype is driven by some objectives that are quite different from those guiding full-scale product implementation efforts.

* A prototype should demonstrate only the minimum set of capabilities that are representative of the design concept. It must be only comprehensive enough to identify potential bottlenecks; for example, a prototype DBMS that performs all operations in main memory would be inadequate. This contrasts with a full-scale implementation that must include an entire complement of features.

* A prototype should be as simple as possible in design and construction. A full-scale implementation may be elegant and/or complex to achieve efficiencies unnecessary in a prototype.

* A prototype provides only a gross proof of technological feasibility. A full-scale DBMS must also satisfy economic and operational feasibility requirements.

* A prototype need not be well designed from a human factors point of view. For a full-scale implementation, ease of use is extremely important.

One objective that applies to prototype and full-scale implementations alike, is that source code should be easy to understand, debug and modify. A DBMS, like any complex software system, should be developed using the best software engineering techniques [FIFE77, BAKE75, BROO75].

For developing a limited DBMS prototype, structured programming practices such as the following are desirable.

* Use of a high-level language.

* Well documented code; liberal use of comments and blank lines.

* Modular design; use of multiple simple, single purpose subroutines rather than large complex procedures.

* Straight-line code; minimum number of branches.

* Single entry and exit for each subroutine.

* All machine and language dependencies isolated in a few well-marked subroutines.

In addition, good coding conventions for the specific source language should be defined and consistently used throughout. The use of modular design and structured coding techniques is especially important because of the experimental purpose for which a prototype DBMS is developed.

<u>3.2.2 Measurement and analysis system.</u> In order to record and evaluate performance characteristics of the prototype software, a measurement and analysis facility is an integral part of a DBMS Performance Prediction System. The measurement facility must be both powerful and flexible. It does not, however, need to consider extremely fine events. A high-level measurement capability can provide data consistent with the coarse performance prediction objectives of a prototype development, measurement, and modeling effort.

For the purpose of evaluating design concepts, a DBMS can be viewed as a facility built on a foundation made up of hardware and operating system components as illustrated in Figure 3.2. While hardware and operating system resource allocation and scheduling procedures are interesting and important determinants of performance for any software operating in the environment they comprise, the focus of this research was the evaluation of DBMS design concepts. To achieve this objective, events can be measured and recorded at the DBMS procedure level rather than at the operating system service request (SVC) level.

DBMS ENVIRONMENT

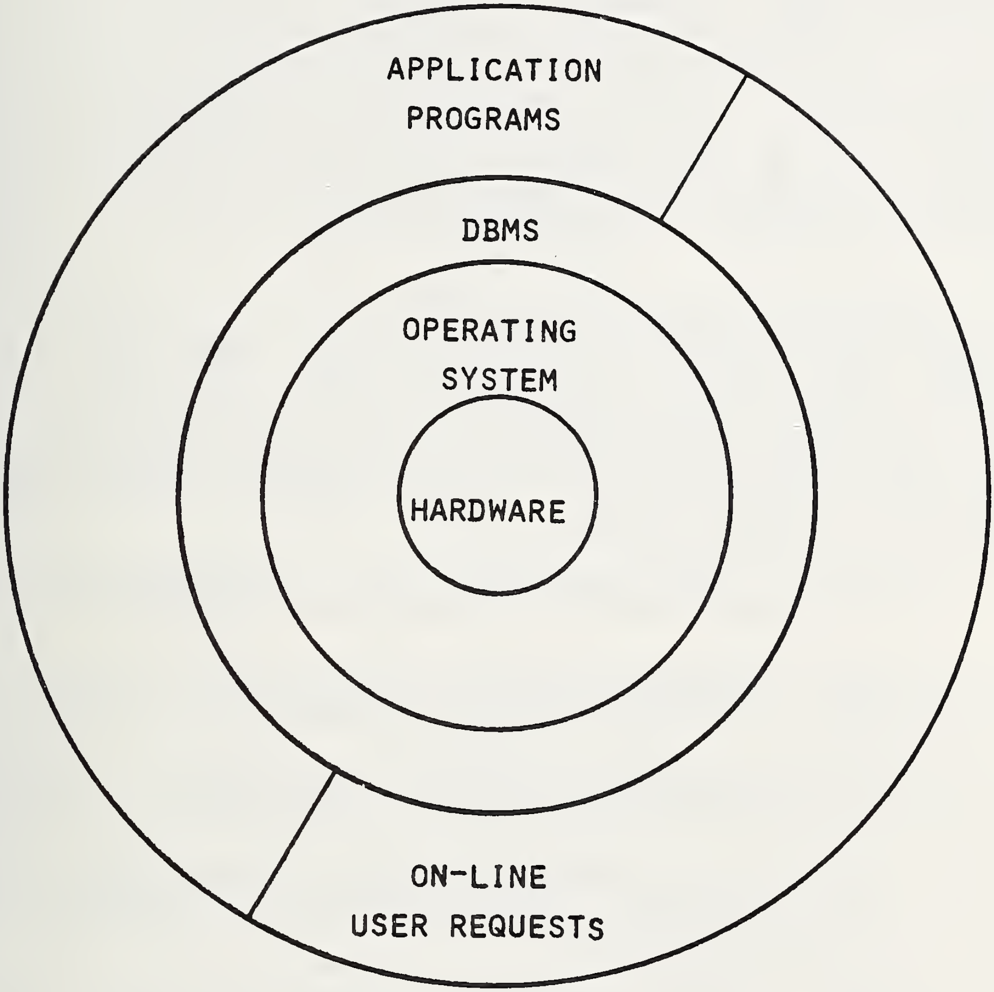


FIGURE 3.2

High-level measurement is desirable for several reasons. First, it can be implemented using a high-level language and does not require modification of the operating system. Consequently, the level of effort is one of man weeks rather than months. High-level measurement does not preclude subsequent instrumentation at the SVC level if deemed necessary. Measuring coarse events is consistent with the limited objectives defined above for the prototype DBMS; it would be foolish to measure low-level indicators of efficiency for software specifically designed without concern for performance. Finally, because full-scale implementations may be installed on hardware other than that used for the prototype, measurement should be independent of specific hardware and operating systems.

<u>3.2.3 Performance prediction model.</u> The third component of an integrated DBMS Performance Prediction System is a performance model that is derived from and can be validated against the prototype using measurement system results. The model should be parameterized so that it not only can estimate performance for the prototype DBMS, but also can consider "what if" questions like the impact on predicted performance of the following.

* What if database sizes were substantially larger than those that the prototype can handle?

* What if database storage structures other than those used by the prototype were employed?

* What if database characteristics such as redundancy and logical complexity were changed?

* What if gross changes in the hardware/operating system environment were made; e.g., if DBMS functions were microcoded?

* What if DBMS functional capabilities were changed; e.g., if new set theoretic operations were added?

While models can be either purely analytic or completely stochastic, because of their nature and complexity, database management systems lend themselves to hybrid (partially analytic with stochastic/simulation components) model representations.

The question of modeling approach is difficult to discuss without considering a specific DBMS object. While there are numerous alternatives, two diametrically opposed approaches represent extreme points of a continuum on which others fall. One method is to develop a high-level conceptualization of a broad class of database management software systems, and then to add more specificity incrementally to the model until the performance of the prototype DBMS can be predicted. This approach is appealing because, from the outset, the model is applicable not just to the concepts of in the prototype, but to a broad class of DBMS designs. There is considerable risk, however, that such a broad conceptualization will not describe the prototype DBMS closely enough to predict its performance accurately; if this occurs, the model can not be validated in even the most limited sense.

At the other extreme is an inductive approach that first develops a model specific to the prototype DBMS, and then incrementally relaxes constraints to produce increasingly general models. The risk associated with this approach is that a specific model may not be easily broadened beyond the prototype system. Two advantages of the inductive approach are the reasonable expectation that the model can be validated, and the assurance that the model will at least address the DBMS design concepts being studied.

In practice, it is unlikely that DBMS model development will occur at either of the above extremes. The difficulties associated with developing general conceptualizations for broad classes of database management systems and the benefits accruing from the ability to validate a prototype specific model are strong inducements for using modeling approaches that fall toward the inductive end of the continuum. Inductive modeling is also required when, as was the case for this research, there is a primary desire to address the performance potential of concepts embedded in the prototype, and only secondarily to consider other DBMS designs.

3.3 Set Processor Performance Prediction System

Figure 3.3 is a graphical representation of the Set Processor Performance Prediction System, a specific implementation of the DBMS development approach described above. The Positional Set Processor (PSP), prototype DBMS incorporates several state-of-the-art capabilities and design concepts. The measurement and analysis system is fully integrated with the prototype DBMS, with measurement capabilities invoked through the DBMS user-interface grammar. Finally, the Set Processor Performance Model (SPPM) includes a model driver and four utility modules as well as size and response time modelers.

Components of the set processor performance prediction system, installed on the PDP-1Ø computer in the National Bureau of Standards' Experimental Computer Facility, are described in the next four chapters. First, the Positional Set Processor prototype DBMS that is the object of the modeling and measurement effort is described. The measurement and analysis system is the subject of the next chapter. Chapters six and seven discuss the concepts and parameters, and the mathematical relationships embedded in the set processor performance model, respectively.

SET PROCESSOR PERFORMANCE PREDICTION SYSTEM OVERVIEW

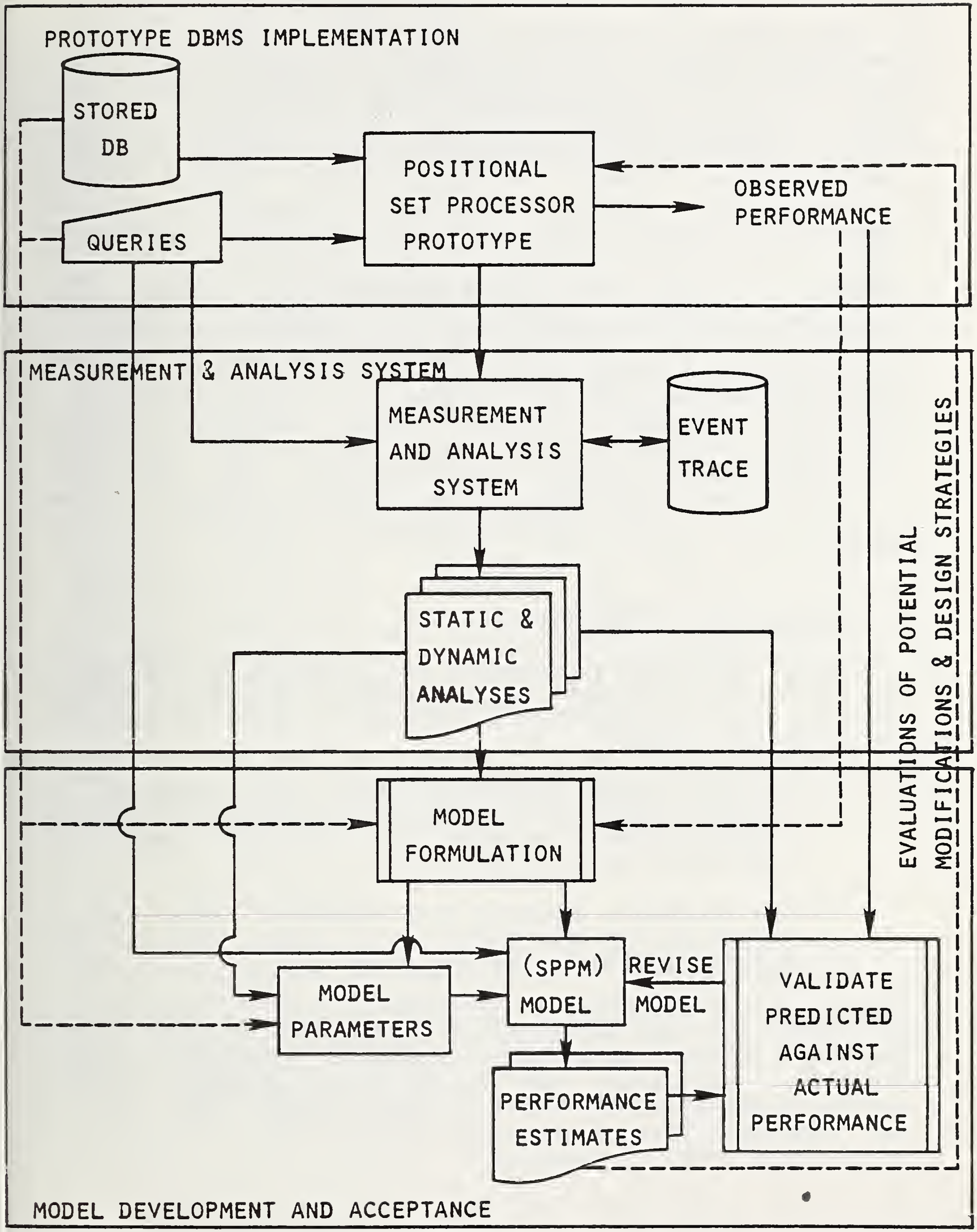


FIGURE 3.3

4. POSITIONAL SET PROCESSOR DBMS PROTOTYPE

The object of the modeling and measurement effort described herein is a prototype for a set-theoretic implementation of a database management system with a relational user interface. The prototype system was conceived and initially implemented by Hardgrave using a PRIME minicomputer at the NASA Institute for Computer Applications in Science and Engineering (ICASE). This preliminary version of the Positional Set Processor (PSP) was transported to the NBS Experimental Computer Facility's PDP-1Ø in the summer of 1976. Described in the literature under the name SET-P [HARD76a-b], the initial ICASE implementation has been substantially enhanced in the NBS ECF testbed. The following sections describe the positional set processor prototype at a level of detail sufficient for understanding the object of this measurement and modeling effort. Readers desiring a more detailed description are directed to the referenced articles and reports. After a brief discussion of the theoretical foundations of the prototype DBMS, PSP design and implementation concepts are considered. The last section describes the status of the prototype when it arrived at NBS, at the time of publication, and planned for the future.

4.1 Theoretical Foundations

Childs proposed an extended set notation for representing both classical sets and (ordered) sequences as collections of value-position pairs [CHIL68a-b, CHIL77]. His contribution is two-fold:

* he uses a level of mathematical rigor uncommon in database management research; the axiomatic definition of Childs' extended set notation appears in [CHIL68b], and

* his framework supports both classical sets and n-tuples at the same definitional level; that is, neither is defined using the other.

Positional set notation, a variation of Childs' extended set representation, was developed by Hardgrave. It captures the power of Childs' work in a somewhat simplified format [HARD77]. Positional set notation can represent sets and sequences nested at any level of complexity; that is, it can represent sets of sequences, sets of sets of sequences, sequences of sets of sequences, ... etc.

The PSP prototype uses positional set notation to implement the relational data model introduced by Codd in 1970 [CODD70] and subsequently reformulated by Hardgrave [HARD78]. Unlike many other relational implementations such as INGRES [STON76], the PSP is a true set processor in that duplicate entries can not occur in a relation. It is generally faithful to the relational data model as mathematically defined by Codd and others [CODD70, CODD71a-b, CODD72, DATE75].

4.2 Design Concepts

Several innovative implementation ideas are embedded in the PSP prototype. One major objective of this research was to evaluate the performance potential of a system using these design concepts. Some of the most important features of the prototype positional set processor implementation are described briefly in the following paragraphs. Much of this material is derived from and the reader is referred to HARD76a-b and HARD73a containing more detailed treatments.

4.2.1 Integer set processor. The functional core of the PSP is an Integer Set Processor (ISP). The current ISP is an improved version of the one previously described by Hardgrave [HARD73a]. Classical sets of positive integers from a predefined universe are represented by compacted bit strings. Traditional set operations (AND, OR, EXCLUSIVE OR, and SET DIFFERENCE) can be performed on these bit strings without uncompacting. Set operations are performed rapidly on existing bit-string representations.

A subset S of the universe of positive integers U (limited only by the word size of the computer) is represented by a logical bit string such that, if we denote the i-th bit of the bit string as i, then:

i = 1 <==> i is an element of S
    0 otherwise

The size of a virtual bit string is the cardinality of (i.e. number of elements in) the universe. A compaction technique must be employed to reduce the size of bit strings so that they can be machine stored and manipulated. The QUATREE compaction method illustrated in Figure 4.1 uses a multiple level hierarchy of n-bit packets. Each bit in a packet at level L corresponds to one n-bit packet at level L-1. A

Set Definition and QUATREE parameters:

S = {4, 11, 25} = Integer Set

L = 16 = Number of Levels for Compaction Tree

n = 4 = Number of Bits in Compaction Packet

Linear Representation:

Bits	1000	1000	...	1000	1100	1010	0001	0010	0010	1000
Level	16	15	...	4	3	2	1	1	2	1

Hierarchical Representation:

```
Level 16                          1000
                                    .
   .                                .
   .                                .
   .
Level 4                           1000
                                   |
                                   |
Level 3                           1100
                                  /  \
                                 /    \
Level 2                       1010     0010
                              /  \        \
                             /    \        \
Level 1                  0001     0010     1000
```

Figure 4.1

zero bit at level L indicates that all bits in the corresponding packet at level L-1 are zero; a 1-bit at level L indicates that at least one of the n bits in the corresponding packet at level L-1 is set to 1. The desired compaction is achieved by omitting all zero packets from the actual bit-string representation.

QUATREE compacted set representations are unique and can be operated on directly by various ISP functions including the classical set operations and set traversal. The latter is a function that enumerates all elements in a set one element at a time. The ISP also provides I/O facilities for transferring bit-string set representations from main memory buffers to secondary storage and vice versa. Figure 4.2 is an annotated list of ISP functions.

In addition to providing the foundation for representing positional sets as described below, the ISP is used by the secondary indexing mechanism added to the PSP in 1977. An index may be viewed as having three components:

* an attribute reference, a;

* an ordered non-redundant list of values, V = <v1,v2,...,vn>, for attribute a; and

* a collection of tuple references or pointers for each value vi, Pi = {pi1,pi2,...,pin}.

Because pointer collections Pi are sets of integer references, the ISP can be used to process boolean queries using indices. Furthermore, many secondary index manipulations can be viewed as set operations that benefit from the availability of a true set processor.

4.2.2 Mapping positional sets onto integer sets. Positional sets are collections of value-position pairs. For the PSP to use the ISP bit-string representation and processing capabilities, a mapping of ordered (value, position) pairs to integers is needed. The PSP uses a diagonal mapping first proposed by Cauchy (1789-1857) and Cantor (1845-1918) to demonstrate that the rational numbers were countable. The Cauchy/Cantor technique is reversible; that is, it is both *one-to-one* and *onto* [HALM64]. Given integers K and L, using the Cauchy/Cantor diagonal method we can calculate a unique integer M.

(K,L) ==> M

Conversely, given M we can calculate K and L.

M ==> (K,L)

FUNCTION REFERENCE OR TYPE	DESCRIPTION
I4UN(IS1,IS2,IS3)	IS3 = IS1 U IS2
I4IN(IS1,IS2,IS3)	IS3 = IS1 $\cap$ IS2
I4RC(IS1,IS2,IS3)	IS3 = IS1 - IS2
I4SD(IS1,IS2,IS3)	IS3 = (IS1 U IS2) - (IS1 $\cap$ IS2) exclusive union; also symmetric difference
I4NULL(IS1)	IS1 = $\emptyset$
I4ONE(k,IS1)	IS1 = {k}
I4UN1(k,IS1)	IS1 = IS1 - {k}
I4COPY(IS1,IS2)	IS2 = IS1
SET TRAVERSAL I4TRVI(...) I4TRV(....) I4TRVE(...)	Set Traversal
BUFFER MANAGEMENT I4ALOC(...) I4SAVE(...) I4DEST(...)	Buffer Management

Figure 4.2

The PSP maps positional sets of value-position pairs to bit-string integer set representations, performs operations on these representations, and maps from the resulting ISP sets back to collections of value-position pairs.

4.2.3 Storage structures. The positional set processor stores and manipulates data using five types of files or tables. Components in the PSP table structure are listed below.

* ELEMENT TABLE - contains non-redundant instances of everything known by the PSP. Fixed length entries include position identifiers (attribute names), atomic elements, and positional sets. The latter are bit-string representations for sets and sequences including tuples (sequences of atom-position pairs), and relations (set of tuples). Storage is non-redundant across the entire database; that means, for example, that the integer 90 could appear in the source database as an age, as an address, and even as a position identifier, but it would appear only once in the element table.

* ELEMENT TABLE INDEX - a fixed length table providing rapid access to the element table. A simple form of hash coding with a linear search end-around strategy is used to locate pointers to element table entries.

* ALIAS TABLE - contains user defined names for referencing element table entries. Names stored in fixed length alias table entries are bound to element table entries by pointers.

* TEXT TABLE - contains overflow from fixed length element and alias table entries. This table has a variable number of variable length records.

* SECONDARY INDEX TABLES - additional tables required for providing rapid access to stored data using secondary indices. These tables contain the three components described above for a secondary index: indicators of which attributes are indexed, an ordered non-redundant list of instances V = <vi> for each attribute, and a set of tuple pointers Pi = {pij} for each attribute value vi. The latter component is represented by one or more tables of bit-string integer set representations that reference element table entry numbers for tuples containing the specified attribute values.

Figure 4.3 illustrates the positional set processor's table structure and depicts relationships among the five components described above.

4.2.4 User interface. The PSP prototype is an interactive query answering system. The user carries out a dialogue with the positional set processor input module developed using an interactive language design system, LANGPAK [HEIN75]. This generalized facility generates a runtime module that performs required input parsing and translation functions and allows easy modification of the language syntax. The relational user's view is currently supported by PSP commands that are consistent with the work of Rothnie [ROTH75] and are, to some extent, similar to language interfaces for relational systems developed in other laboratories [STON76, CHAM76, ASTR76, CODA62].

4.2.5 Modular high-level code. The positional set processor is made up of approximately 200 subroutines written in ANSI FORTRAN. The measurement system and ISP functions that are replicated by PSP subroutines add another 70 programs to this total. Machine dependencies, notably I/O and bit manipulation, are isolated in a very few subroutines. The transportability of the code has been demonstrated by its installation in three substantially different environments: the ICASE PRIME minicomputer, the University of Maryland UNIVAC 1108, and the NBS PDP-10.

The entire software system is highly modular and reflects good structured design and coding practices [FIFE77, KERN74, MILL76]. Common block definitions are specified in separate files that are referenced via the FORTRAN "INCLUDE" statement by PSP subroutines. Procedures have single entries and exits, are liberally commented and have an abundance of white space for ease of reading. The source listings for most subroutines require only one printed page.

4.3 Capabilities: Past, Present and Future

The positional set processor implementation has been and remains a true prototype in that the desire for simplicity rather than for speed and/or efficiency has guided implementation decisions. Care has been taken not to preclude an efficient implementation, however. Some comments about the capabilities of the PSP prototype in the past, at the current time, and projected for the future follow.

POSITIONAL SET PROCESSOR TABLE STRUCTURE

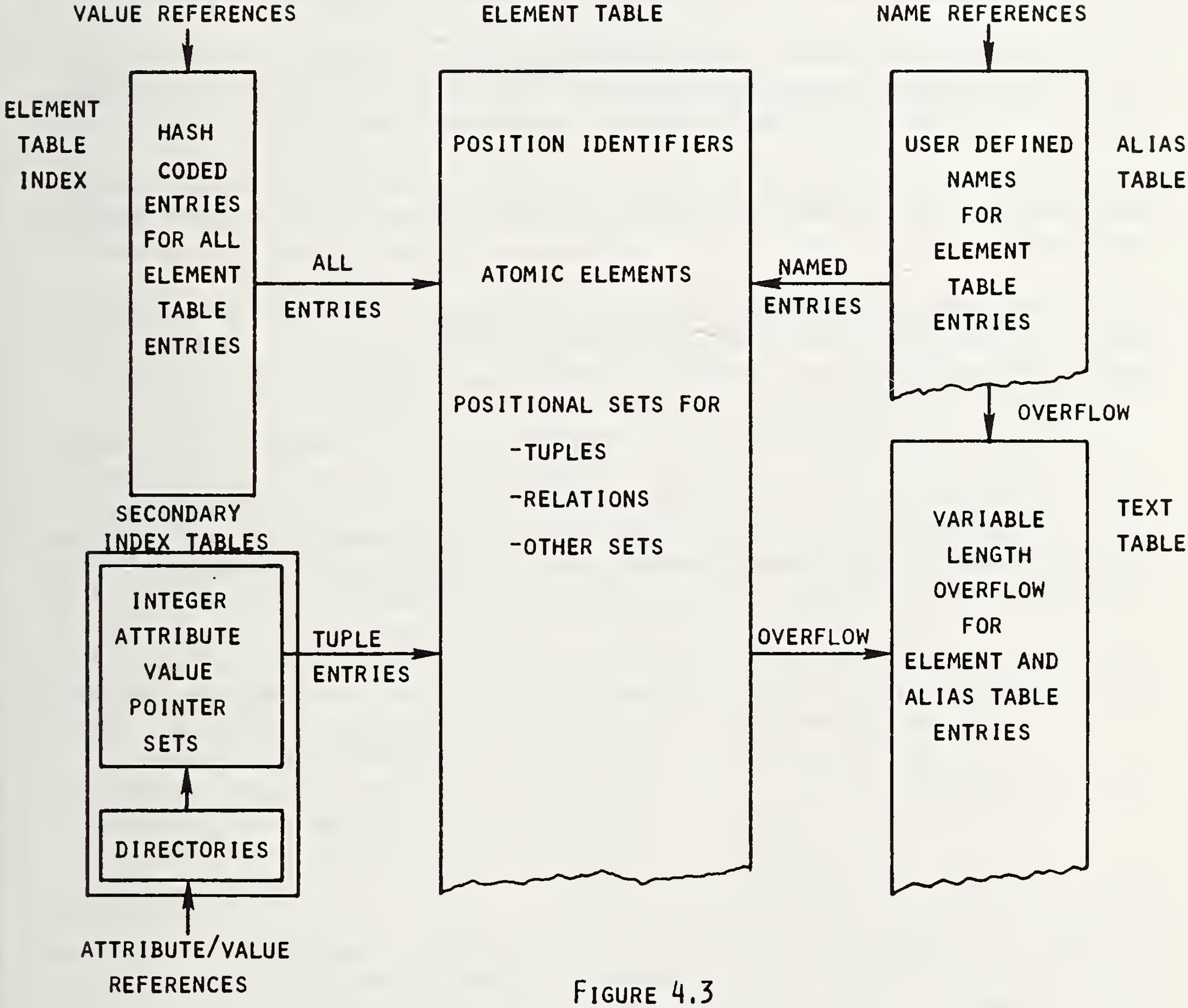


FIGURE 4.3

4.3.1 Past capabilities. The positional set processor prototype that was transported to the NBS PDP-1Ø in the summer of 1976 had several limitations.

* System response was extremely slow for many operations, even for queries on very small relations. This was caused by gross implementation inefficiencies and by the lack of a secondary indexing mechanism.

* Databases loaded using the prototype had been limited to single relations of fewer than 2Ø tuples.

* Data definition was very crude, requiring reentry of a file containing domain characteristics and formatting information whenever data was read in or out.

In spite of these limitations, the prototype DBMS demonstrated the most basic level of feasibility; the implementation could store, retrieve and manipulate classical sets.

4.3.2 NBS enhancements: current status. Since its installation in the NBS ECF testbed, the PSP prototype has been substantially enhanced. Improvements include the following.

* A secondary indexing mechanism was added to allow selection by value without having to enumerate exhaustively all tuples in a relation. The index facility uses the integer set processor to store, retrieve and manipulate sets of tuple pointers for each value of an indexed attribute.

* System response time was improved, both by the addition of secondary indices and by correction of some gross implementation inefficiencies. There never was a desire to optimize system performance, but rather to make the prototype a more useful laboratory tool. Changes in the prototype were guided by early insights and results derived from the measurement and modeling efforts.

* Database sizes were increased by approximately two orders of magnitude. In addition to the original very small (7-15) tuple database, databases of 56 and 5ØØ tuples have been loaded. All databases use secondary indices.

* Additional capabilities were added to the PSP to assist the user and to make it a more useful experimental tool. Many changes were small, but some (e.g. commands for modifying automatic buffer allocations) had substantial impact on the prototype's flexibility and viability.

The current version of the positional set processor is clearly improved; it runs faster, handles more data, and does more things than the ICASE prototype from which it evolved. It is still a limited laboratory prototype, however. Some of the most obvious deficiencies are listed below.

* There still is no comprehensive data definition capability. Until one is implemented, the positional set processor can not be considered a complete database management system.

* Response time could be improved further especially for non-indexed queries.

* The cost in processor and elapsed time for loading the largest, 5ØØ tuple database is too great. Time for loading and building indices will have to be reduced before the next order of magnitude for database size can be achieved.

* While there are no theoretical or known practical restrictions, the prototype still has not been used for databases with more than one relation (multiple relations in the current version are all derived from a single initially loaded relation).

* Secondary indices are static; that is, database changes are not reflected in the indices.

* A range variable capability is missing; one is needed to handle multi-structure queries.

<u>4.3.3 Future plans.</u> It is expected that the positional set processor will be the object of increasing study and improvement efforts. In addition to applying results derived from the measurement and modeling effort, the major deficiencies of the current implementation listed above will be addressed. Two interesting possibilities for future research are the use of hardware for selected PSP functions and/or a major recoding effort to consolidate subroutines and to make the PSP more efficient. Decisions to pursue these or other research directions will be made using the set processor measurement system and performance model described below.

5. SET PROCESSOR MEASUREMENT AND ANALYSIS SYSTEM

In order to develop a set processor model, a measurement system was needed for providing insight into the operation of the prototype, and for validating model generated performance predictions. Alternative approaches were considered including measurement of operating system service requests (SVC's). This traditional approach to measurement through the use of operating system level software probes was seen as having several disadvantages and limitations for database modeling. Measurement of database systems at ICASE, carried out at the SVC level, required substantial levels of effort [SHER76]. The set processor modeling project goal of determining gross feasibility for proposed DBMS designs did not require micro-level measurements. Finally, experiments with the NBS PDP-1Ø clock indicated that it was too coarse for low-level event measurement. Consequently, a decision was made to measure prototype performance at a high level consistent with the project goals, manpower and hardware resources.

The positional set processor measurement system installed in the NBS Experimental Computer Facility testbed is illustrated in Figure 5.1. Written entirely in FORTRAN (as is the PSP prototype), the event recorder and analysis programs allow selective recording and reporting of gross performance statistics. Events are defined at the FORTRAN program level; that is, the measurement system records processor and clock times for execution of selected prototype DBMS programs. In addition, disk I/O's are monitored and trace records indicating type, magnitude and specific references for each secondary storage access can be generated. All processor and clock times are adjusted to remove measurement system caused perturbations.

5.1 Event Recorder

The PSP event recorder allows the user selectively to accumulate execution times and (optionally) to record event traces. Measurement requests can be changed at any time during the execution of the prototype DBMS. The PSP user interface grammar accepts measurement system commands interspersed in the PSP instruction stream. Thus, the interactive user can turn on the measurement system after preparing an experiment, exercise the set processor capability or component that is of interest, and then turn off and/or change parameters for the event recorder without leaving the realm of positional set processor control.

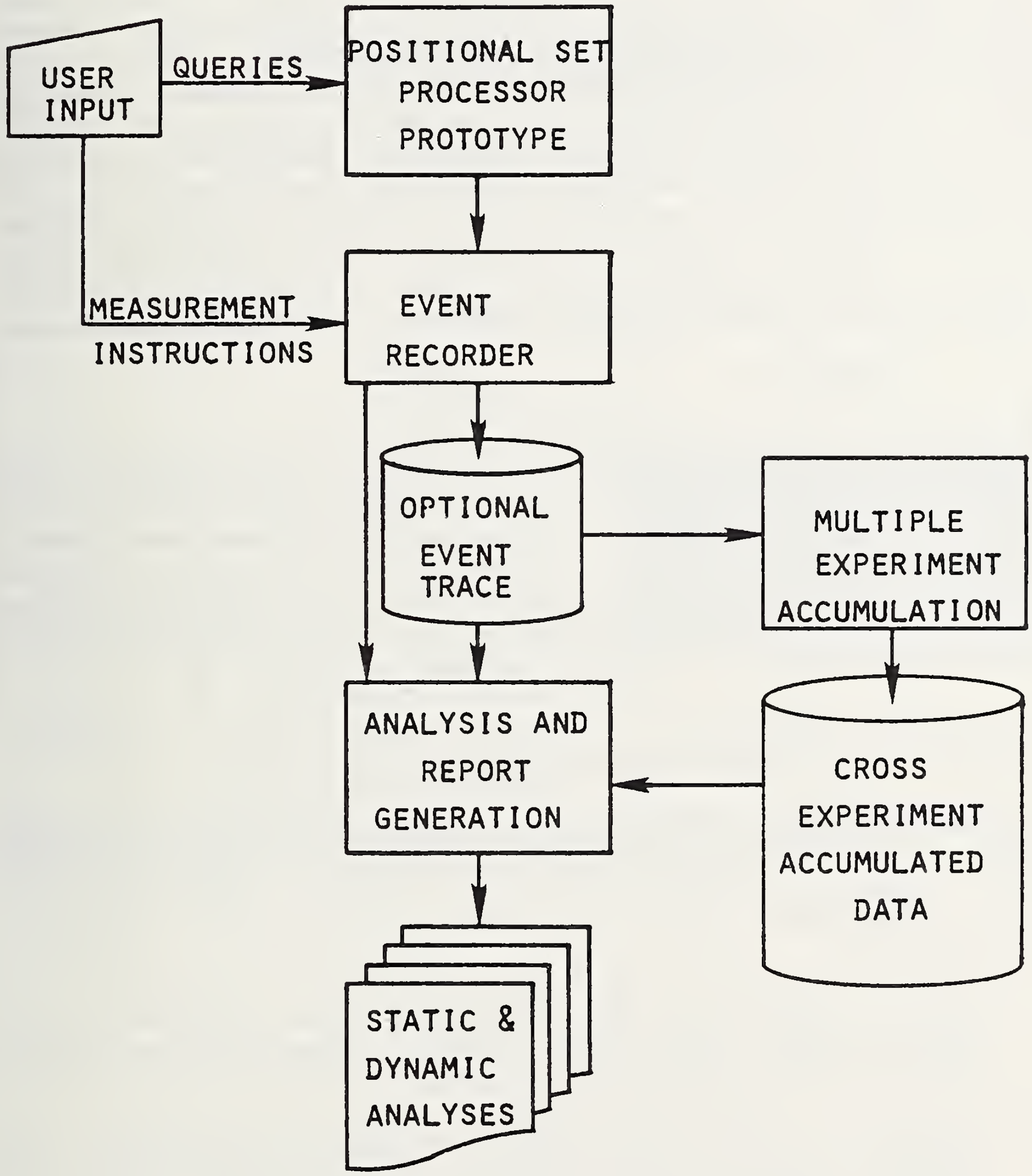

USER
INPUT
QUERIES
POSITIONAL SET
PROCESSOR
PROTOTYPE
MEASUREMENT
INSTRUCTIONS
EVENT
RECORDER
OPTIONAL
EVENT
TRACE
MULTIPLE
EXPERIMENT
ACCUMULATION
ANALYSIS AND
REPORT
GENERATION
CROSS
EXPERIMENT
ACCUMULATED
DATA
STATIC &
DYNAMIC
ANALYSES

FIGURE 5.1

The user of the event recorder has the flexibility of requesting data about specific PSP programs, groups of programs determined by leading characters in program names, and program "trees". The last option allows measuring all activity from the time a specified program is entered until that program is completed; data for the "tree" of all subroutines called in the interim is thus recorded.

Two types of measurement can be requested: gross measures of total times between program entry and exit events, and detailed recording of event trace records. Gross measures provide only static time summaries; that is, times for a program include those for the tree of all programs called between entry and exit of that program. Event trace records provide the necessary detailed data required to produce mutually exclusive time analyses for measured programs. Trace records are tightly packed 72 bit (two word) representations written in 128 word physical records, corresponding to the NBS PDP-10 basic disk I/O unit (sector) size.

5.2 Multiple Experiment Accumulation

While event trace files are produced for specific experimental sessions, measured results can optionally be aggregated across experiments. The multiple experiment accumulation component of the measurement and analysis system aggregates CPU and clock times in permanent secondary storage files. Aggregation may occur over a number of experiments and over an extended period or time.

5.3 Analysis and Report Generation

A comprehensive analysis and report generation facility provides tools for processing and outputting measurement results. Figure 5.2 lists the six reports currently produced by PSP measurement analysis programs and classifies them as follows.

* Static - gross accumulations of times between program entry and exit events generated on the fly without writing trace records.

* Dynamic - time and event summaries derived from trace files that recognize chronological event sequences and can thus determine mutually exclusive program times.

POSITIONAL SET PROCESSOR
MEASUREMENT AND ANALYSIS SYSTEM

ANALYSIS AND REPORT GENERATION

	SINGLE SESSION	AGGREGATE
STATIC	Static Summary	
DYNAMIC	Formatted Path Dynamic Summary Trace Event Listing Transition Report	Aggregate Summary

Figure 5.2

* Single Session - reports representing a single positional set processor session or portion thereof.

* Aggregate - reports representing accumulation of measurement system results across multiple positional set processor sessions.

Each of the six output reports is described briefly in the paragraphs below.

5.3.1 Static summary. Figure 5.3 is an illustrative copy of the static summary report. The report includes: replications of user query inputs; counts for calls to the measurement system (referenced by the FORTRAN program name SVC4ØØ), event trace records written, and logical and physical I/O references; CPU and clock times between beginning and end of measurement session for processing and for measurement (the latter denoted ISVCPU and ISVCLK); and line items summarizing activity for each measured program. Programs are listed, in descending order, based on the total amount of CPU time required. Counts for the number of program entries and for I/O reads, writes and deletes appear for each program. Level indicators refer to the positions within the tree of all programs invoked during the session where the program appeared; the lowest level (1) corresponds to the root, while the higher the level number the closer a program was to the "leaves" of the program tree. Totals, averages and dispersions are presented for both CPU and clock times. Dispersions, serving as indicators of relative variability, are calculated by dividing standard deviations by mean times.

This is a static analysis because program CPU and clock times are not mutually exclusive, but rather represent totals for all activities occurring between program entry and exit. The static summary report is produced whenever measurement occurs, regardless of whether an event trace is recorded.

5.3.2 Formatted path. The formatted path analysis report is illustrated in Figure 5.4. Generated from event trace records, this report lists program names in their order of execution. A left to right, top down indentation and line spacing strategy indicates calling sequences and program dependencies. I/O events appear as special records with the format:

<Operation> <Number of SAU's>/<File Name>/<Starting SAU>

This report is an extremely useful visual footprint path for prototype DBMS program executions.

STATIC SUMMARY REPORT

```
TURN ON GROUP M9;

TURN ON TREE K9IO;

TURN ON TRACE USING DEMO;

TURN ON TIME;

SUBX *A=VGH DATA BASE(X.(*):X.NAME.EQ.TWA);

EXIT;


SVC400 SUMMARY

NUMBER OF CALLS TO SVC400 =        712

NUMBER OF 72 BIT EVENTS WRITTEN TO DEMO       =     380

    L9RD=    43    L9WT=     5    L9DE=     0

    K9 WORDS =        3380         L9 SECTORS =        73

TIME BETWEEN TRACE FILE OPENING AND CLOSING

      CPU=    8369    CLOCK=   35324    ISVCPU=    3368    ISVCLK=    5291
```

PROG	#ENT	CPU TOT	CPU AVG	CPU DISP	CLK TOT	CLK AVG	CLK DISP	L9RD	L9WT	L9DE	LEVEL HI	LEVEL LO	
9	2	8362	4181.0	0.9353	35314	17657.0	0.9361	43	5	0	1	1	M9COMM
5	2	7883	3941.5	0.9700	14402	7201.0	0.7274	43	5	0	2	2	M9PROC
241	43	6528	151.8	0.5968	11186	260.1	1.2987	43	5	0	4	3	K9IO
243	48	5983	124.6	0.5780	10524	219.3	1.4311	43	5	0	5	4	L9IO
239	48	1602	33.4	0.8968	3555	74.1	3.4440	0	0	0	6	5	L9OPEN
293	1	1600	1600.0	0.0000	3064	3064.0	0.0000	11	0	0	3	3	M9PTRN
291	2	305	152.5	0.2365	335	167.5	0.2069	0	0	0	3	3	M9ADD1
79	5	52	10.4	0.7020	86	17.2	1.0347	0	0	0	4	3	M9GETS
289	1	29	29.0	0.0000	40	40.0	0.0000	0	0	0	3	3	M9PIDS
73	5	22	4.4	0.2591	39	7.8	0.5080	0	0	0	4	3	M9GETL
7	1	15	15.0	0.0000	24	24.0	0.0000	0	0	0	3	3	M9ALOC
297	2	7	3.5	0.2020	8	4.0	0.0000	0	0	0	3	3	M9ISOL

Figure 5.3

FORMATTED PATH ANALYSIS REPORT

```
PATH ANALYSIS OF TRACE DATA FILE  DEMO2

M9COMM    TNOUA     TNOU
          M9PROC    M9GETL
                    M9GETL
                    M9GETS    **  2
                    M9PIDS    M9GETS
                              M9GETL
                    M9ALOC    M2ALOC
                    M9PTRN    M9GETS
                              A7TRAN    A7SRCH    A3GETE    K9IO      L9IO      L9OPEN
                                                                                R  2Ø/57A/    6
                                                  A3GETE    K9IO      L9IO      L9OPEN
                                                                                R  2Ø/57A/   26
                                                  A3GETE    K9IO      L9IO      L9OPEN
                                                                                R  2Ø/57A/   46
                                                  A3GETE    K9IO      L9IO      L9OPEN
                                                                                R  2Ø/57A/   66
                                                  A3GETE    K9IO      L9IO      L9OPEN
                                                                                R  2Ø/57A/   86
                                                  A3GETE    K9IO      L9IO      L9OPEN
                                                                                R  2Ø/57A/  1Ø6
                                                  A3GETE    K9IO      L9IO      L9OPEN
                                                                                R  2Ø/57A/  126
                                                  A3GETE    K9IO      L9IO      L9OPEN
                                                                                R  2Ø/57A/  146
                                                  A3GETE    K9IO      L9IO      L9OPEN
                                                                                R  2Ø/57A/  166
                              M9GETL
                              M9GETS
                              M7REGA    N7PUT     N5FIND    N5HASH
                                                            N5IO      K9IO      L9IO      L9OPEN
                                                                                          R 44Ø/55E/ 2311
                                                            N3GETE    K9IO      L9IO      L9OPEN
                                                                                          R 34Ø/57E/  946
                                                            STREQ
                    A7TRAN    A7SRCH    A3GETE    K9IO      L9IO      L9OPEN
                                                                      R  2Ø/57A/    6
```

Figure 5.4

5.3.3 Dynamic summary. The dynamic summary report, illustrated in Figure 5.5, has a format similar to that for the body of the static summary described above. Program line item entries contain mutually exclusive times in the dynamic analysis report, however. That is, CPU and clock times for any one program do not include processing times for other measured subroutines.

5.3.4 Trace event summary. A portion of a trace event summary report is reproduced in Figure 5.6. This output contains one line for each program entry/exit event appearing in a trace file; I/O events are ignored. The "Entry #" column contains ordinals for entry/exit event pairs within all executions of a specific program. Pseudo times, time increments since previous events, and elapsed times between entries and exits are reported for both CPU and wall clocks. Because of the large number of trace records that are generated for even a relatively limited query sequence, the trace event summary report is generated only on occasions when detailed incremental measurement results are required.

5.3.5 Transition report. The measurement analysis facility notes when time changes can not be attributed to measured programs and records these apparent anomalies on a non-zero transition time listing. Non-zero transition time can result from execution of procedures not selected for measurement, and from measurement system malfunctions.

5.3.6 Aggregate summary. The aggregate summary report, having a format similar to that for the single session dynamic summary time analysis, lists accumulated CPU and clock times for PSP programs across multiple experiments.

5.4 Using a Coarse Clock to Measure Fine Events

Virtually all modern computers provide hardware clocks that can be accessed through system software. While these facilities are generally adequate for controlling operating system functions and for time accounting, they sometimes are inadequate for monitoring the execution of even complete procedure sequences of machine instructions [WORT76, GENT73]. Timing difficulties can be attributed to many factors. Gentlemen and Wickman [GENT73] discuss a number of timing problems related to the increasing use of complex multiprogramming operating systems and multilevel storage hierarchies.

In a multiprogramming environment there is no simple relationship between elapsed time on a hardware clock and processing time for any given task. The NBS PDP-1Ø employs a technique that is widely used for timing separate

DYNAMIC SUMMARY REPORT

TIME ANALYSIS OF TRACE DATA FILE DEMO

		CPU			CLK			
PROG	ENTRY	TOT	AVG	DISP	TOT	AVG	DISP	NAME
243	4Ø	381Ø	95.3	Ø.72Ø5	5715	142.9	1.Ø836	L9IO
239	4Ø	1349	33.7	Ø.9698	3269	81.7	3.4182	L9OPEN
5	2	493	246.5	1.3969	563	281.5	1.3991	M9PROC
9	2	475	237.5	Ø.6275	2Ø9Ø6	1Ø453.Ø	1.Ø545	M9COMM
241	39	445	11.4	Ø.4765	549	14.1	Ø.3493	K9IO
291	2	3Ø5	152.5	Ø.2365	335	167.5	Ø.2Ø69	M9ADD1
293	1	135	135.Ø	Ø.ØØØØ	3Ø8	3Ø8.Ø	Ø.ØØØØ	M9PTRN
79	5	52	1Ø.4	Ø.7Ø2Ø	86	17.2	1.Ø347	M9GETS
73	4	19	4.8	Ø.2Ø16	34	8.5	Ø.4945	M9GETL
289	1	18	18.Ø	Ø.ØØØØ	17	17.Ø	Ø.ØØØØ	M9PIDS
7	1	15	15.Ø	Ø.ØØØØ	24	24.Ø	Ø.ØØØØ	M9ALOC
297	2	7	3.5	Ø.2Ø2Ø	8	4.Ø	Ø.ØØØØ	M9ISOL
SUM		7123			31814			
TRANSITION		4			5			
TOTAL		7127			31819			

Figure 5.5

TRACE EVENT SUMMARY REPORT

ANALYSIS OF TIME INCREMENTS FOR FILE DEMO

STATUS	PROG	ENTRY#	CPU	CPU INC	CPU ELAPSED	CLK	CLK INC	CLK ELAPSED
ENTER	M9COMM	1	66364	0		50525340	0	
ENTER	M9PROC	1	66661	297		50543775	18435	
ENTER	M9GETL	1	66666	5		50543783	8	
LEAVE	M9GETL	1	66671	5	5	50543792	9	9
ENTER	M9GETL	2	66676	5		50543799	7	
LEAVE	M9GETL	2	66680	4	4	50543806	7	7
ENTER	M9GETS	1	66686	6		50543813	7	
LEAVE	M9GETS	1	66692	6	6	50543821	8	8
ENTER	M9GETS	2	66698	6		50543830	9	
LEAVE	M9GETS	2	66721	23	23	50543879	49	49
ENTER	M9PIDS	1	66726	5		50543891	12	
ENTER	M9GETS	3	66732	6		50543898	7	
LEAVE	M9GETS	3	66737	5	5	50543907	9	9
ENTER	M9GETL	3	66743	6		50543914	7	
LEAVE	M9GETL	3	66749	6	6	50543928	14	14
LEAVE	M9PIDS	1	66755	6	29	50543931	3	40
ENTER	M9ALOC	1	66761	6		50543941	10	
LEAVE	M9ALOC	1	66776	15	15	50543965	24	24
ENTER	M9PTRN	1	66781	5		50543978	13	
ENTER	M9GETS	4	66784	3		50543983	5	
LEAVE	M9GETS	4	66794	10	10	50543993	10	10
ENTER	K9IO	1	66804	10		50544006	13	
ENTER	L9IO	1	66809	5		50544015	9	
ENTER	L9OPEN	1	66815	6		50544022	7	
LEAVE	L9OPEN	1	66839	24	24	50544068	46	46
LEAVE	L9IO	1	66901	62	92	50544172	104	157
LEAVE	K9IO	1	66906	5	102	50544178	6	172
ENTER	K9IO	2	66915	9		50544189	11	
ENTER	L9IO	2	66921	6		50544199	10	
ENTER	L9OPEN	2	66926	5		50544208	9	
LEAVE	L9OPEN	2	66948	22	22	50544236	28	28
LEAVE	L9IO	2	67010	62	89	50544339	103	140
LEAVE	K9IO	2	67015	5	100	50544349	10	160
ENTER	K9IO	3	67025	10		50544360	11	
ENTER	L9IO	3	67030	5		50544369	9	
ENTER	L9OPEN	3	67036	6		50544379	10	
LEAVE	L9OPEN	3	67059	23	23	50544414	35	35

Figure 5.6

multiprogramming tasks; a logical (software) clock runs (i.e., is incremented) only when its associated task is being executed. Wortman [WORT76] lists several kinds of undesirable variability in timing information resulting from this approach including the following.

* There may be a variable delay between the starting and/or stopping of a task and the corresponding posting of the logical (software) clock.

* The operating system may be sometimes arbitrary in charging time to tasks, e.g. "short" interrupts may be charged to the interrupted task rather than to the task that caused the interrupt.

* A variable delay may occur between a request to the operating system for time information and the actual reading of the clock.

* Charging time for I/O operations is complex and often arbitrary.

In addition to demonstrating all of the above listed problems, the NBS PDP-1Ø clock has a granularity that is derived from the cycle rate of the main power supply. Thus, clock increments are measured in 1/6Øth of a second units termed JIFFIES. To further complicate matters, time is recorded as an integer number of milliseconds; a 16 2/3 millisecond JIFFY must be posted as either 16 or 17 milliseconds.

A method for using a coarse timer that was attributed to Sutcliffe and verified by Gentleman and Wickman appears in the appendix of their note. When an event occurs, it involves the use of a tight loop to inspect the clock until it changes. Then, multiplying the number of iterations by the time required to read the clock yields the time that elasped between the event and the new clock time. Figure 5.7 illustrates the following variables associated with this method for using a coarse timer.

t(a) = system clock reading when i-th event occurs.

t(i) = actual time when i-th event occurs; this time can not be determined directly from the coarse system clock.

t(b) = system clock reading at next closest tick; i.e. time immediately following next clock change.

VARIABLES FOR MEASUREMENT USING A COARSE CLOCK

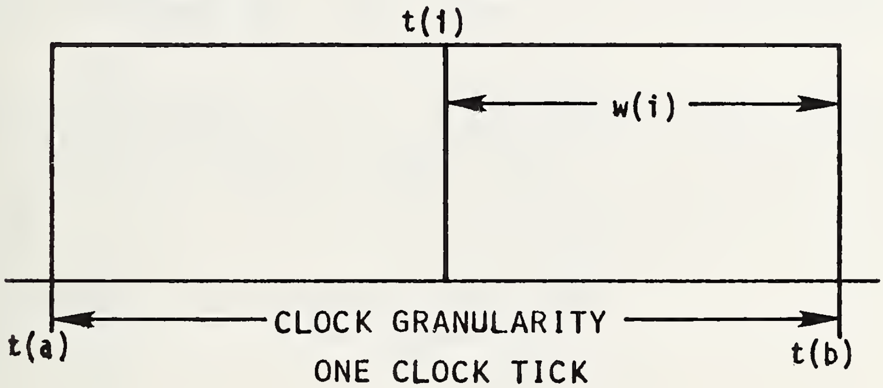


FIGURE 5.7

w(i) = elasped time between i-th event and next clock change; estimated by iterative clock cycling procedure described above.

The Set Processor measurement system uses this costly but viable clock cycling strategy. Because PSP events correspond to subroutine entry and exit points, the measurement system causes a distribution of procedures in relation to system clock time like that illustrated in Figure 5.8. When a subroutine is entered, the measurement systems postpones processing until a clock tick (change) occurs; thus, all procedures start at the beginning of a clock interval. When an exit event indicates that a subroutine is completed, the measurement system again "cycles" until the system clock changes. A substantial amount of processor time is sacrificed to achieve the additional accuracy. Actual measurement cost relative to processing time is a function of the average processor time between events. Specific implementation characteristics of the PSP measurement system are described in the following paragraphs.

5.4.1 Pseudo time calculation. Pseudo times are recorded for positional set processor events. These times have the effect of removing gaps caused by measurement system cycling. Pseudo times are calculated using system clock readings and accumulated cycle times.

tw(i) = total cycle times for all events prior to and including i

$$= \sum_{j=1}^{i-1} w(j) + w(i)$$

$$= \sum_{j=1}^{i} w(j)$$

t'(i) = pseudo time for i-th event

= t(b) - tw(i)

Pseudo times both preserve event chronology and allow calculation of times for PSP procedures with an accuracy of $\pm$ 1 to $\pm$ 1.25 milliseconds.

DISTRIBUTION OF PSP PROCEDURES FROM CLOCK CYCLING

PROGRAM CALLING SEQUENCE

EVENT SEQUENCE

E1. ENTER A
E2. ENTER B
E3. LEAVE B
E4. ENTER C
E5. ENTER C
E6. LEAVE A

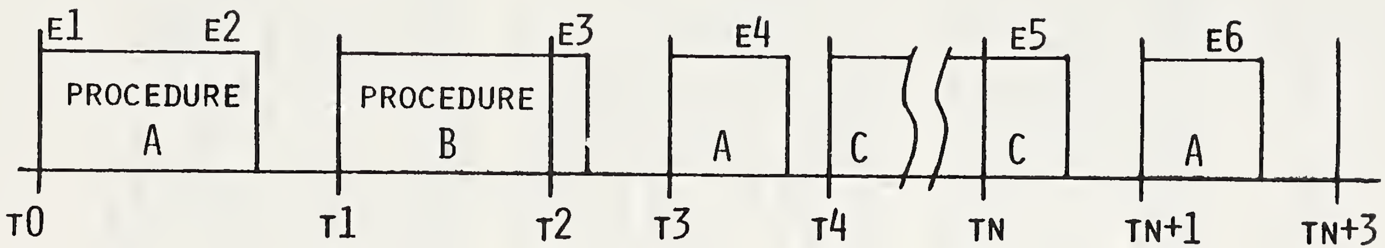


FIGURE 5.8

5.4.2 Cycle calibration. The measurement approach described above is predicated on the assumption that the time required to read the clock is known. If the time is not known, a "chicken and the egg" problem arises; one must use the coarse clock for calibrating the clock enhancement mechanism. This problem exists for the NBS PDP-1Ø. Furthermore, it is complicated by the arbitrary posting of whole millisecond approximations for JIFFY intervals.

Calibration for the PSP measurement system was accomplished by exercising the cycling mechanism thousands of times under various system loads and at all hours of the day and night. Clock increments and numbers of calls were recorded. Least square regression lines were then fitted to the resulting points. Figure 5.9 is a graphical representation of the kinked relationship that resulted from this effort. The kink reflects the arbitrary integer postings when whole 16 2/3 millisecond JIFFY intervals occur.

5.4.3 Quiescent system timing. The Sutcliffe measurement approach assumes that there is no multiprogramming. Acceptable program time dispersion factors have been obtained using the PSP measurement facility for experiments on a multiprogrammed system. This indicates that measurements are consistent within the experiment. However, time magnitudes for specific procedures vary depending on system load. Consequently, for determining parameter values and validating model predictions, measurements are performed on a quiescent system; that is, when no demands (including operating system task scheduling) are imposed on the system other those made by the prototype database system.

5.4.4 Synchronizing processor and wall clocks. The PSP measurement and analysis system records and reports both processor and wall clock times. The cycling strategy can be applied to one clock or the other, but not to both. Experiments with the NBS PDP-1Ø showed that both processor and wall clocks were being incremented at (approximately) the same time. Furthermore, because a task can lose control of the processor at a clock tick, wall clock time may be posted several times for a single processor time increment. Thus, the primary clock for cycling purposes is the processor clock; wall clock anomalies are recognized and adjusted by the measurement system.

CALIBRATION OF CLOCK CYCLING MECHANISM

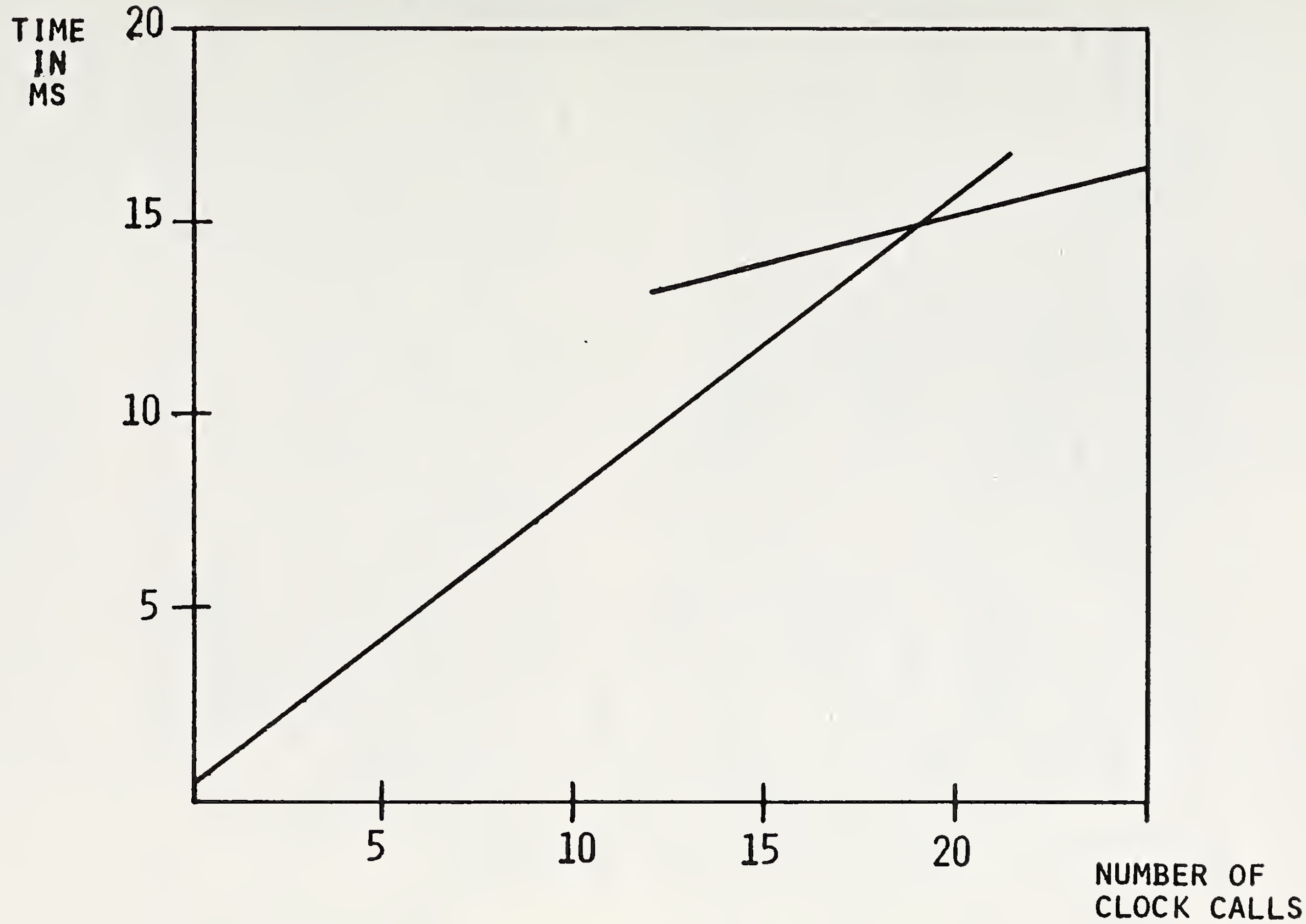


FIGURE 5.9

6. PERFORMANCE MODEL - CONCEPTS AND PARAMETERS

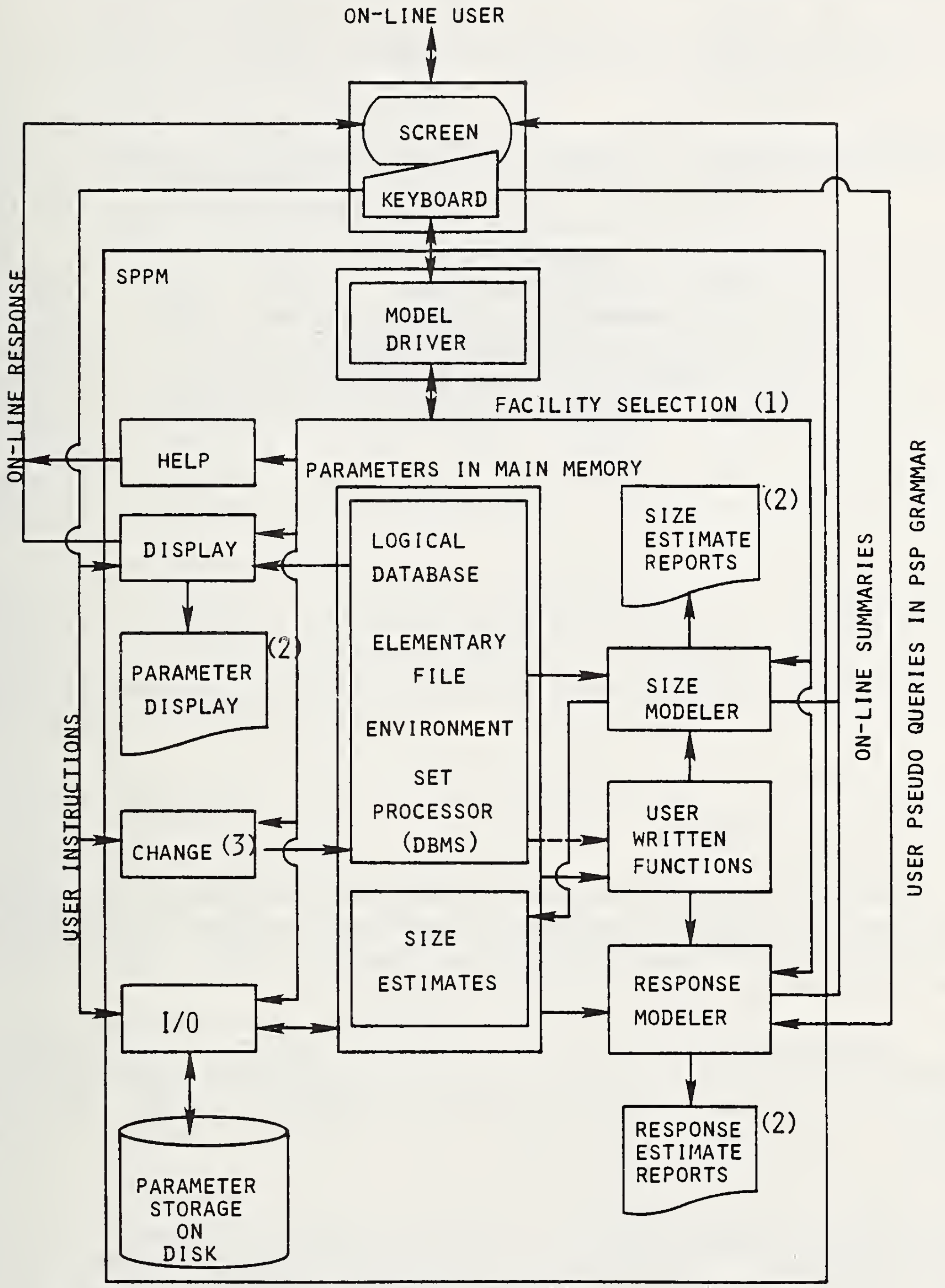
6.1 Introduction

The Set Processor Performance Model (SPPM) is an interactive system for estimating gross indicators of performance potential for the Positional Set Processor and other database management systems incorporating similar design concepts. Written in FORTRAN-10, the SPPM is installed on the NBS Experimental Computer Facility's PDP-10 computer with a KA10 processor and 256K 36-bit words of main memory. Figure 6.1 presents a functional overview of the SPPM system; this figure is described throughout this chapter. Made up of over 200 highly modular subroutines, the Set Processor Performance Model has two major performance prediction modelers, four utility modules and a model driver. Model parameters and derived values describe pertinent application, DBMS and environmental characteristics. The following sections address the model driver, the four utility modules, model parameters, and conceptual foundations for size and response time estimation modelers.

6.2 Model Driver

The model driver is an executive routine that invokes SPPM utility modules and size and response modelers upon request from the on-line user. Except in the case of catastrophic error, control is always returned to the model driver. Presence of the driver is indicated by the unique input request prompt, M>. In response, the user enters one of the eight single letter commands listed and described in Figure 6.2. The HELP, DISPLAY, and CHANGE utilities are invoked by the letters, H, D and C respectively. The I/O module responds to (L)oad and (S)ave commands. The letters Z and R are used to request the SI(Z)E and (R)ESPONSE modelers. Finally, X causes a normal termination or E(X)IT.

SET PROCESSOR PERFORMANCE MODEL (SPPM) - FUNCTIONAL OVERVIEW



(1) Control always returned to model driver.
(2) Hard copy reports written to disk for spooling.
(3) Preliminary implementation of change facility generates display file for modification with editor and subsequent reentry using tabular input facility.

FIGURE 6.1

SET PROCESSOR PERFORMANCE MODEL (SPPM) COMMANDS

```
SPPM COMMANDS - SELECT FROM FOLLOWING LETTERS:
    H = HELP, PRINT THIS SUMMARY
    L = LOAD NEW PARAMETER SET FROM DISK FILE
    D = DISPLAY CURRENT PARAMETER SET
    C = CHANGE PARAMETER SET
    S = SAVE CURRENT PARAMETER SET ON DISK
    Z = RUN SIZE ESTIMATION MODELER
    R = RUN RESPONSE TIME ESTIMATION MODELER
    X = EXIT, TERMINATE EXECUTION OF MODEL

M>
```

Figure 6.2

6.3 Utility Modules

Utility modules perform parameter manipulation and user assistance functions that support and facilitate the use of the size and response modelers. Each of the four utility functions is briefly described below.

6.3.1 (H)elp module. The help module merely displays the command letters and definitions, exactly as they appear in Figure 6.2, at the users on-line terminal. Like the other utility functions, the help module can be invoked whenever the model driver has control.

6.3.2 (D)isplay module. The display module produces on-line and hard-copy displays of SPPM parameters. When invoked, the display facility asks the user to specify whether the annotated parameter listing should be displayed on-line or whether a file for subsequent output on a high-speed line printer should be produced. In either case, listings of parameter values are associated with FORTRAN variable names and are preceded by brief definitions. Note that only parameter values stored in main memory are displayed; parameters maintained on secondary storage and derived size estimates are not accessible through the display facility. Displayed values that are not actually stored as parameters, such as INDEX occurrence counters, are flagged with an asterisk (*) in the parameter display. The initial SPPM implementation does not allow the user to display only a portion of the entire parameter set. It also does not provide for invoking the display module from within the performance prediction modelers. Both of these capabilities would be desirable enhancements.

<u>6.3.3 (C)hange module.</u> The purpose of the change module is to allow the user to modify parameter sets stored in main memory. Ideally, this would be an interactive facility for scanning, extracting and changing parameter values. Such a capability would require access to a modern text editor or similar software from within the model. Because this could not easily be done on the ECF PDP-10 under the TOPS-10 operating system, the preliminary SPPM implementation provides an alternative to interactive parameter modification.

A temporary mechanism for changing existing parameter sets is provided through the generation of a formatted parameter listing stored on disk that can be modified using an on-line text editor, and then reloaded using a tabular input processor. To use the temporary change mechanism, the user follows this scenario:

1. from within the SPPM, generate a formatted parameter listing;

2. exit from SPPM control;

3. use a system provided text editor to modify the formatted parameter display file;

4. reenter the SPPM; and

5. load the modified parameter set using the tabular input facility.

The parameter listing produced in Step 1 has the same format as that generated by the SPPM (D)isplay facility described above. The listing generator as well as the tabular input mechanism can be invoked from within the temporary change facility. As with the display module, only model parameters stored in main memory are addressed by the change facility; derived size estimates are not accessible through this function.

6.3.4 (S and L) Parameter I/O module. The parameter I/O module provides for the storage and retrieval of both SPPM parameters and derived size estimates on secondary storage. The (S)ave command writes parameters and size estimates from SPPM main memory common blocks to a user specified disk file. The process is reversed by the (L)oad command. Storage on disk is in the form of a simple bit stream. This comprehensive I/O capability allows the user, after generating size estimates for a parameter set and using the save facility, to subsequently load saved parameters and invoke the response modeler without again having to run the size modeler.

6.4 SPPM Parameters

Parameters for the Set Processor Performance Model are stored in main memory (FORTRAN) common. Figure 6.3 lists and describes the contents of five common definition files that are invoked by SPPM programs using the FORTRAN "INCLUDE" capability. Each of these files defines one or more blocks of named common storage and lists and describes variable names used for referencing parameters. These five collections of parameter variables and their conceptual foundations are discussed in the following paragraphs.

6.4.1 (LOGCOM) Logical database description. Content and structural characteristics of stored databases are important determinants of DBMS performance [LOWE68, CARD73, CARD75]. The SPPM requires specification of these characteristics in terms of the relational logical view that is assumed for the user. Parameters describe the database; its relations, attributes and domains; and the mapping of attributes onto relations.

In order to minimize the number of parameter definitions required to define a database and to facilitate perturbations of existing logical database descriptions, replication variables are provided for relations and for attributes within relations. These variables allow the user to indicate multiple occurrences of an attribute or relation entity in the database, without having to define separate names and characteristics for each occurrence. For instance, the CLASS relation can be defined with a replication factor of three. This does not mean that there are three CLASS relations in the database, but rather that there are two unnamed relations with the same redundancy, degree and cardinality characteristics as the CLASS relation. Replicated entities are used by the SPPM for estimating size and response times, but cannot be referenced in surrogate queries that drive the response modeler. The replication feature is especially useful for considering the impact of

SUMMARY OF COMMON BLOCK DEFINITION FILES

FOR SPPM PARAMETERS AND RESULTS

FILE NAME	DESCRIPTION	PARAMETERS/RESULTS
LOGCOM	Logical DB Description	database relations attributes domains attribute/domain mappings
FILCOM	Elementary File DB Description	elementary files logical entries
PHYCOM	Physical Characteristics	hardware/software environment
SOFCOM	DBMS Software	DBMS software functions
SPRCOM	Set Processor DBMS	bit-string parameters
ZESCOM	Derived Size Estimates	source DB size stored DB size elementary file and logical entry sizes
RESCOM	Derived Response Estimates	core workspace buffers internal set representation processor times I/O statistics

Figure 6.3

database size on DBMS performance. Once a logical database definition kernel is defined, database size can be easily varied by modifying relation and attribute replication factors and by changing relation cardinalities.

One type of SPPM parameter describing database contents that has not been considered in other DBMS modeling efforts is redundancy. Redundancy or its obverse, uniqueness, is specified at three logical levels in the SPPM parameter set: for the entire database, for relations, and for attributes. The latter is accomplished through the specification of the number of unique values in the domains over which attributes are defined.

Redundancy can be loosely defined in terms of unique instances versus possible occurrences as follows.

$$\text{REDUNDANCY} = 1 - \frac{\text{\# Unique Instances}}{\text{\# Possible Occurrences}}$$

For a relation, this concept includes redundancy across as well as within its attributes. Similarly, at the database level, redundancy considers all relations in the database. For certain secondary storage utilization strategies, redundancy is an important determinant of database size. To the degree that I/O impacts performance, redundancy indirectly can effect response time as well.

Figure 6.4 is a copy of the LOGCOM common definition file including complete annotated listings of SPPM variables for describing logical database structure and content. This and other parameter descriptions have the following columns.

* PARAMETER - FORTRAN variable names.

* DESCRIPTION - brief description of parameter.

* DEF - indication of storage class for parameter; e.g. I = integer, F = floating point, A3 = 3 characters alphabetic.

* BUF SUB - ordinals for PDP-10 words indicating the position of each variable relative to the beginning of the common block.

XXFIL and XXSEC variables provide space for the addition of new parameters and pad to disk sector (128 words) boundaries, respectively.

```
C***********************************************************
C
C        LOGCOM - COMMON BLOCK REFERENCE
C
C        PARAMETERS DESCRIBING LOGICAL DB CHARACTERISTICS
C
C***********************************************************
C
C   - DATABASE LEVEL PARAMETERS
C
         COMMON/LDBCOM/DBNAM,DBRDN,DBNRL,DBNDM,DBNAT,DBNUA,DBFIL(122)
C
         INTEGER DBNAM,DBRDN,DBNRL,DBNDM,DBNAT,DBNUA,DBFIL
C
C      PARAMETER       DESCRIPTION                      DEF    BUF SUB
C
C      _________   ___________________________         ___    _______
C
C         DBNAM    NAME OF DB                           A5        1
C         DBRDN    REDUNDANCY % OVER ALL REL            I         2
C         DBNRL    NO. OF REL DEFS IN DB                I         3
C         DBNDM    NO. OF DOMAIN DEFS IN DB             I         4
C         DBNAT    NO. OF ATTRIBUTE DEFS IN DB          I         5
C         DBNUA    NO. OF UNIQUE ATTRIBUTES IN DB       I         6
C         DBFIL    FOR FUTURE USE                       _         7 - 128
C
C   - RELATION LEVEL PARAMETERS
C
         COMMON/LRLCOM/RLNAM(100),RLRPL(100),RLRDN(100),RLDEG(100),
      1                     RLCRD(100),RLFIL(100,3),RLSEC(96)
C
         INTEGER RLNAM,RLRPL,RLRDN,RLDEG,RLCRD,RLFIL,RLSEC
C
C      PARAMETER       DESCRIPTION                      DEF    BUF SUB
C
C      _________   ___________________________         ___    _______
C
C         RLNAM    NAME OF RELATION                     A5        1 - 100
C         RLRPL    NO. OF REPLICATIONS FOR REL          I         101-200
C         RLRDN    REDUNDANCY % OVER ALL ATTS.          I         201-300
C         RLDEG    DEGREE = NO. OF ATT IN REL           I         301-400
C         RLCRD    CARDINALITY = NO. OF TUPLES          I         401-500
C         RLFIL    FOR FUTURE USE                       -         501-800
C         RLSEC    FILL TO SECTOR BOUNDARY              -         801-896
C
C   - ATTRIBUTE PARAMETERS
C
         COMMON/LATCOM/ATNAM(100),ATDOM(100),ATFIL(100,3),ATSEC(12)
C
         INTEGER ATNAM,ATDOM,ATFIL,ATSEC
```

Figure 6.4a

```
C
C     PARAMETER        DESCRIPTION                        DEF    BUF SUB
C
C     _________   ___________________________          ___    _______
C
C        ATNAM    NAME OF ATTRIBUTE                   A5       1 - 1ØØ
C        ATDOM    NAME OF DOMAIN FOR ATTRIBUTE        A5       1Ø1-2ØØ
C        ATFIL    FOR FUTURE USE                      -        2Ø1-5ØØ
C        ATSEC    FILL TO SECTOR BOUNDARY             -        5Ø1-512
C
C   - DOMAIN PARAMETERS
C
         COMMON/LDMCOM/DMNAM(1ØØ),DMNVL(1ØØ),DMAVS(1ØØ),DMFIL(1ØØ,3)
      1                    DMSEC(4Ø)
C
         INTEGER DMNAM,DMNVL,DMAVS,DMFIL,DMSEC
C
C     PARAMETER        DESCRIPTION                        DEF    BUF SUB
C
C     _________   ___________________________          ___    _______
C
C        DMNAM    NAME OF DOMAIN                      A5       1 - 1ØØ
C        DMNVL    NO. OF VALUES IN DOMAIN             I        1Ø1-2ØØ
C        DMAVS    AVG SIZE IN BITS OF DOM INST        I/*A4    2Ø1-3ØØ
C        DMFIL    FOR FUTURE USE                      -        3Ø1-6ØØ
C        DMSEC    FILL TO SECTOR BOUNDARY             -        6Ø1-64Ø
C
C   - PARAMETERS MAPPING ATTRIBUTES AND RELATIONS
C
         COMMON/LRACOM/RAINM(1ØØ,1ØØ),RASEC(112)
C
         INTEGER RAINM,RASEC
C
C     PARAMETER        DESCRIPTION                        DEF    BUF SUB
C
C     _________   ___________________________          ___    _______
C
C        RAINM    REL/ATT INCIDENCE MATRIX            I        1-1Ø,ØØØ
C
C        WHERE: RAINM(I,J) = K
C
C                 |K| = NUMBER OF REPLICATIONS OF JTH ATTRIBUTE
C                         IN THE ITH RELATION
C
C                  K  <  Ø  =>  JTH ATTRIBUTE IN ITH RELATION
C                                 IS INDEXED
C
C                  K  >  Ø  =>  JTH ATTRIBUTE IN ITH RELATION
C                                 IS NOT INDEXED
C
C        RASEC    FILL TO SECTOR BOUNDARY             -        1ØØØ1-1Ø112
C
C
C****END OF LOGCOM COMMON BLOCK REFERENCE**************************
C
```

Figure 6.4b

6.4.2 (FILCOM) Elementary file description. Another determinant of DBMS performance is its utilization of secondary storage. Among others, Senko et. al. have pursued the problem of mapping (logical) data structures onto (physical) storage devices by identifying multiple hierarchical abstract levels falling between logical structures and their physical representations. Senko's DIAM II, for example, has a five level hierarchy [SENK73]. While there may be disagreement about the generality of the structures employed (Senko uses a string representation) and the number of levels falling between the logical and physical extremes, the concept of a continuum of abstractions is well established. Figure 6.5 is a pictorial representation of the use of multiple conceptual levels for mapping information onto physical storage.

The LOGCOM parameters described above address the portion of the continuum labeled "data structures" in the figure. Storage structures are described by parameters in common definition file FILCOM. These SPPM parameters describe secondary storage structures in terms of elementary files and their logical entries.

In the landmark FOREM and follow-on PHASE-II modeling work done by Senko, Owens and others, logical data was mapped onto hardware devices using data sets consisting of one or more elementary files each with its own physical record format [SENK70, OWEN71]. The SPPM uses the term "elementary file" in a slightly different but related way. An elementary file (EF) is defined as a collection of logical entry (LE) occurrences. Elementary files may contain many logical entry types, each with different characteristics; each LE type occurs in only one named EF, however.

FILCOM parameters span the secondary storage portion of the continuum in Figure 6.5. In general, elementary file representations are at a level above the physical representation device and I/O software specific end of the continuum. Physical representation aspects are addressed, however, by parameters for LE and fixed length EF sizes, for I/O software main memory buffer sizes, and for overhead at both the logical entry and elementary file levels. The mappings of data structures onto storage structures are encoded in SPPM parameters that specify EF and LE occurrence frequencies in terms of logical (LOGCOM) entries. A functional description of secondary storage is used for classifying logical entries. These and other concepts embodied in the FILCOM parameters listed in Figure 6.6 are discussed below.

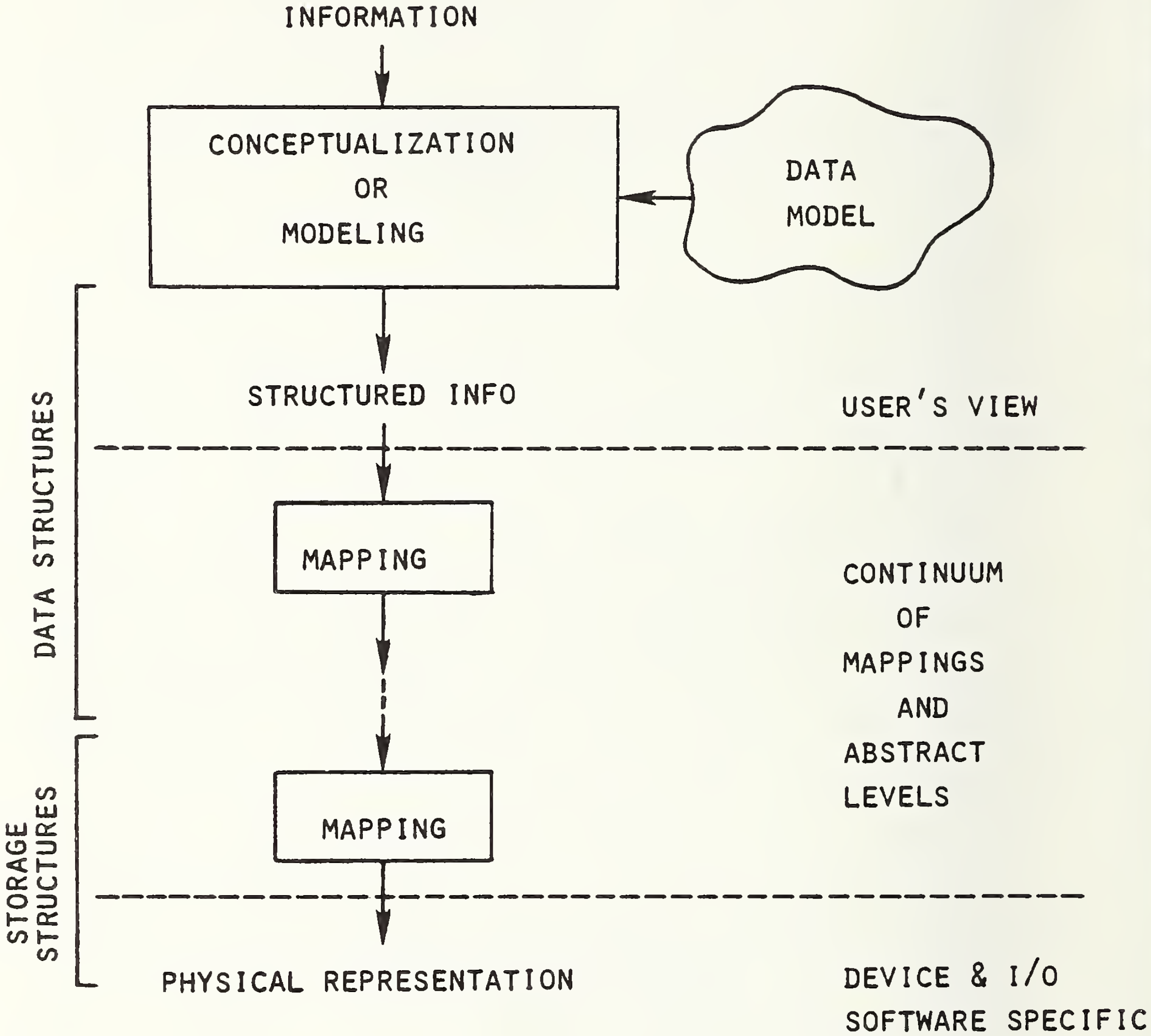

INFORMATION
CONCEPTUALIZATION
OR
MODELING
DATA
MODEL
STRUCTURED INFO
USER'S VIEW
DATA STRUCTURES
MAPPING
CONTINUUM
OF
MAPPINGS
AND
ABSTRACT
LEVELS
STORAGE
STRUCTURES
MAPPING
PHYSICAL REPRESENTATION
DEVICE & I/O
SOFTWARE SPECIFIC

FIGURE 6.5

```
C*************************************************************
C
C       FILCOM - COMMON BLOCK REFERENCE
C
C       ELEMENTARY FILE AND RELATED PARAMETERS
C
C************************************************************
C
C   - GLOBAL FILE PARAMETERS
C
        COMMON/FGFCOM/GFNEF,GFNLE,GFFIL(126)
C
        INTEGER GFNEF,GFNLE,GFFIL
C
C    PARAMETER       DESCRIPTION                   DEF    BUF SUB
C
C    _________   ___________________________      ___    _______
C
C       GFNEF    NO. OF EF DEFINITIONS IN
C                 PARAMETER SET                    I        1
C       GFNLE    NO. OF LE TYPES DEFINED FOR
C                 ALL EF'S IN PARAMETER SET        I        2
C       GFFIL    FOR FUTURE USE                    -        3 - 128
C
C   -  ELEMENTARY FILE PARAMETERS
C
        COMMON/FEFCOM/EFNAM(100),EFLET(100),EFIXZ(100),EFFOH(100),
     1            EFEOH(100),EFNOC(100),EFRFO(100),EFRFQ(100),
     2            EFBUF(100),EFFIL(100,2),EFSEC(52)
C
        INTEGER EFNAM,EFLET,EFIXZ,EFFOH,EFEOH,EFNOC,EFRFO,EFRFQ,
     1            EFBUF,EFFIL,EFSEC
C
C    PARAMETER       DESCRIPTION                   DEF    BUF SUB
C
C    _________   ___________________________      ___    _______
C
C       EFNAM    NAME OF EF                        A5       1 - 100
C       EFLET    NO. OF LOG ENTRY TYPES IN EF      I        101-200
C       EFIXZ    FIXED SIZE FOR EF (INCLUDES
C                 ALL LE'S AND ALL OVERHEAD)       I/*A4    201-300
C       EFFOH    O.H. IN BITS FOR FILE             I/*A4    301-400
C       EFEOH    O.H. IN BITS FOR EACH LE          I/*A4    401-500
C       EFNOC    NO. OF OCCURENCES OF EF FOR
C                 SPECIFIED RFO AND RFQ            I        501-600
C       EFRFO    REL ENTITY DET EF OCCURENCE       A4       601-700
C       EFRFQ    QUALIFIER FOR RFO                 A5       701-800
C       EFBUF    I/O SOFTWARE BUFFER SIZE          I        801-900
C                 IN BIOU'S
C       EFFIL    FOR FUTURE USE                    -        901-1100
C       EFSEC    FILL TO SECTOR BOUNDARY           -        1101-1152
C
```

Figure 6.6a

```
C   - PARAMETERS FOR EF LOGICAL ENTRIES
C
      COMMON/FLECOM/LENAM(100),LEFUN(100),LEFRF(100),LENOC(100),
     1               LERFO(100),LERFQ(100),LESIZ(100),LEOHD(100)
     3               LEFIL(100,3),LESEC(52)
C
C    PARAMETER      DESCRIPTION                         DEF    BUF SUB
C
C    _________   ___________________________           ___    _______
C
C       LENAM    NAME OF LOGICAL ENTRY                 A5       1 - 100
C       LEFUN    FUNCTIONAL TYPE FOR LE                A5       101-200
C       LEFRF    ELEMENTARY FILE REF                   A5       201-300
C       LENOC    NO. OF OCCURENCES OF LE
C                 FOR SPECIFIED RFO AND RFQ            I        301-400
C       LERFO    REL ENTITY DET LE OCCURENCE           A4       401-500
C       LERFQ    QUALIFIER FOR RFO                     A5       501-600
C       LESIZ    AVERAGE SIZE IN BITS OF LE            I/*A4    601-700
C       LEOHD    O.H. IN BITS FOR LE OCCURENCE         I/*A4    701-800
C       LEFIL    FOR FUTURE USE                        -        801-1100
C       LESEC    FILL TO SECTOR BOUNDARY               -        1101-1152
C
C***END OF FILCOM COMMON BLOCK REFERENCE***************************
C
```

Figure 6.6b

Parameter functions Most SPPM parameters are simple variables; that is, user provided parameter values contain all of the information required by the model. Another class of specification can be used for several FILCOM parameters. Parameter functions allow the user to specify relationships that do not lend themselves to description with simple variables. Two types of parameter functions are used by the SPPM: intrinsic functions for describing LE and EF occurrence frequencies, and optional functions that provide a mechanism for invoking special user written FORTRAN procedures that determine secondary storage characteristics.

* INTRINSIC FUNCTIONS - occurrence frequencies for logical entries and elementary files are specified by parameter triples of the form:

  XXNOC times for each XXREO in XXREQ

  where:
  XX = EF or LE
  NOC = integer
  REO = relational entity occurrence indicator
  REQ = relational entity qualifier

  For instance, a logical entry might be defined as occurring:

  <2> times for each <TU>ple in relation <PEREL>

  Intrinsic functions perform (sometimes complex) procedures to determine occurrence frequencies from parameter tuples. Because relational entity occurrence indicators (REO's) and relational entity qualifiers (REQ's) refer to entity types and specific entity names defined in LOGCOM parameters, intrinsic functions map logical data structures onto elementary file representations for storage structures. Furthermore, this mapping means that even when FILCOM parameters are held constant, EF and LE occurrence frequencies can change when LOGCOM logical database descriptions change.

* OPTIONAL FUNCTIONS - when simple parameter constants are not sufficient for describing secondary storage structures, the SPPM allows the user to invoke FORTRAN procedures denoted by placing an asterisk (*) followed by an up to four character function names in the parameter set. The parameters for which optional functions can be defined are described with a

"/*A" in the format column of the common definition file annotation. Optional functions provide flexibility and generality for representing complex and/or unique storage structures with SPPM parameters.

Functional description A useful representation of secondary storage structures should both provide insight into the utilization of storage resources and assist in evaluating alternative strategies. To achieve these objectives, the SPPM elementary file description of secondary storage requires that the non-overhead portion of each logical entry be associated with a specific secondary storage function. A useful taxonomy for secondary storage structures must be comprehensive; that is, it should describe a large portion of existing and proposed DBMS storage utilization strategies. The four-part functional taxonomy illustrated in figure 6.7 meets this criteria and provides the necessary insights.

Most existing systems do not specifically segment secondary storage structures into four distinct components for representing data, primary relationships, secondary relationships and definition. They can be easily described, however, in these terms. Furthermore, to the degree that these elements are not explicitly recognized as unique and separable, database design tradeoffs can be made without sufficient consideration for their impact on storage and processing costs.

Brief descriptions for each of these four functional components of secondary storage structures follow.

* DATA - data instance storage. The size relative to other components can vary greatly depending on the accessing strategies employed, the definition of atomic elements (e.g. field values vs. records), and the handling of redundancy.

* PRIMARY (INTRINSIC) RELATIONSHIPS - relationships among data that are derived directly from the (logical) data structure. The amount of storage required varies from none (when all intrinsic relationships are signified by physical contiguity) to multiples of that required for data storage (when all intrinsic relationships are explicitly represented with pointers, lists, etc.). Regardless of how they are represented, intrinsic relationships cannot be discarded; together with data, they form a complete (although possibly inefficient for data accessing) physical representation of the structured information.

FUNCTIONAL TAXONOMY OF SECONDARY STORAGE STRUCTURES

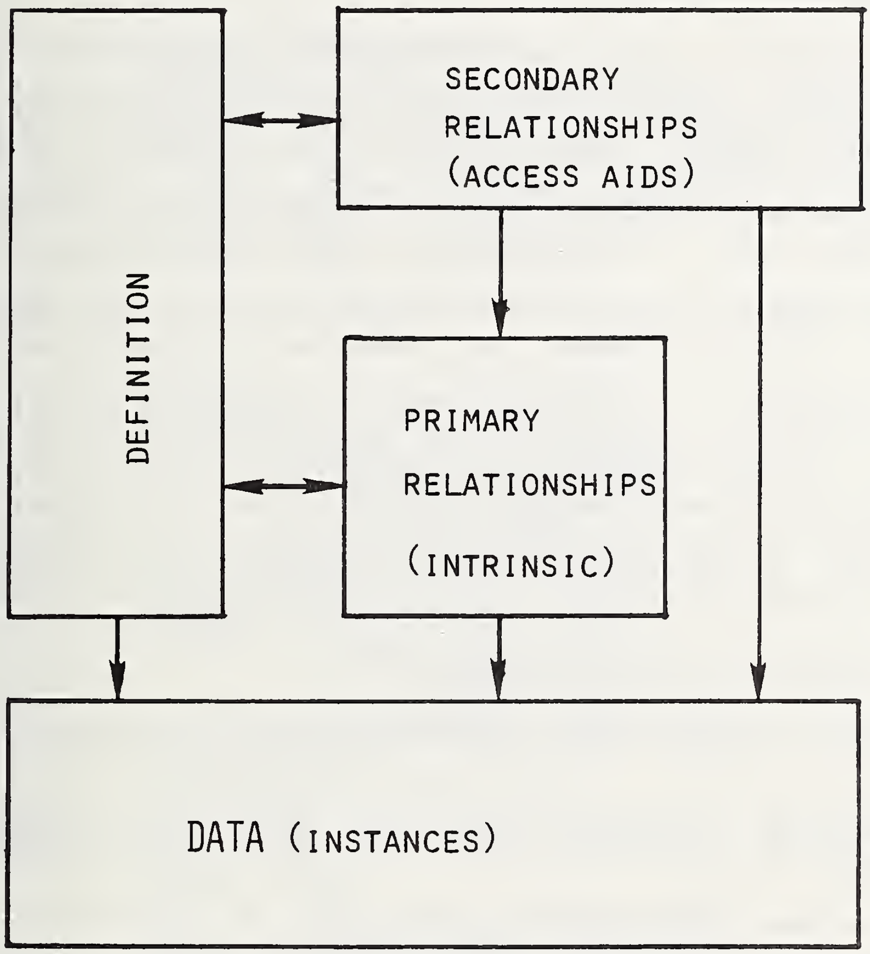


FIGURE 6.7

* SECONDARY RELATIONSHIPS (ACCESS AIDS) - relationships among data that are not derived directly from the data structure, but are defined in order to expedite accessing of the stored data. Because of their existence solely for increasing efficiency, these storage structure components can be destroyed without altering the completeness of the physical representation.

* DEFINITION - descriptive entries required for identifying, encoding and decoding physical representations on secondary storage. This component includes data element descriptors such as size (number of bits, characters, words, fields, etc.), class (alphabetic, numeric, etc.), mode (integer, floating point, etc.), and reference names and synonyms.

Many existing and proposed DBMS storage structures can be described in terms of these four functional categories.

<u>Overhead</u> Usually it is not possible to attribute all of the secondary storage required for logical entries and their elementary files to the four functions described in the previous section. Requirements for additional, non-functional secondary storage are considered overhead by the SPPM. Overhead for a particular FILCOM parameter set is derived from parameters describing four overhead classes:

* elementary file overhead,

* excess of fixed EF size over computed size for all LE's in an EF,

* overhead associated with all LE's in a specific named EF, and

* overhead for a specific LE type.

Overhead calculations can also be influenced by optional functions specified for FILCOM parameters.

<u>6.4.3 (PHYCOM) Physical environment.</u> SPPM parameters describing the physical environment for the object DBMS appear in the Figure 6.8 listing of the annotated PHYCOM definition file. Physical environment parameters fall in three categories: hardware characteristics, system load, and I/O times.

<u>Hardware characteristics</u> Hardware architecture is described in terms of smallest addressable units (SAU's) and basic I/O units (BIOU's). The term "smallest addressable unit" refers to the amount of main memory that is generally accessed; it

```
C*****************************************************************
C
C       PHYCOM - COMMON BLOCK REFERENCE
C
C       PHYSICAL CHARACTERISTICS
C
C*****************************************************************
C
C   - HARDWARE/SOFTWARE ENVIRONMENT
C
        COMMON/PENCOM/ENPPI,ENLOD,ENSAU,ENCPS,ENBIU,ENIOM,
     1  ENIRA,ENIWA,ENIRT,ENIWT,ENIOP,ENICL,ENIDE,ENFIL(115)
C
        INTEGER ENSAU,ENCPS,ENBIU,ENIOM,ENIRA,ENIWA,ENIRT,ENIWT,
     1           ENIOP,ENICL,ENIDE,ENFIL
C
C    PARAMETER        DESCRIPTION                      DEF    BUF SUB
C
C    _________  ____________________________          ___    _______
C
C       ENPPI   PROCESSOR POWER INDICATOR              F        1
C                WHERE: 1=NBS ECF PDP/10
C                       2=MACHINE WITH TWICE
C                         PDP-10 PROCESSOR
C                         POWER(SPEED)
C                     .5=1/2 PDP-10 SPEED
C       ENLOD   COEF OF SYSTEM LOAD                    F        2
C                WHERE: 1=QUIESCENT SYTEM
C                         DEDICATED TO PSP
C                       N>1 => REAL TIME = N x
C                         QUIESCENT SYSTEM TIME
C       ENSAU   SIZE IN BITS OF SMALLEST               I        3
C                ADDRESSABLE UNIT (SAU)
C       ENCPS   NUMBER OF CHARACTERS PER SAU           I        4
C       ENBIU   SIZE IN SAU'S OF BASIC I/O             I        5
C                UNIT (BIOU)
C       ENIOM   MAX NO. OF BIOU'S TRANSFERRED          I        6
C               WITH ONE ACCESS(SEEK + LATENCY)
C       ENIRA   ACCESS TIME IN MILLISECS FOR           I        7
C                SECND STORAGE READ
C       ENIWA   ACCESS TIME IN MILLISECS FOR           I        8
C                SECND STORAGE WRITE
C       ENIRT   TRANSFER TIME IN MILLISECS FOR         I        9
C                SECND STORAGE READ
C       ENIWT   TRANSFER TIME IN MILLISECS FOR         I        10
C                SECND STORAGE WRITE
C       ENIOP   TIME IN MILLISECS TO OPEN SECND        I        11
C                STORAGE FILE
C       ENICL   TIME IN MILLISECS TO CLOSE SEC         I        12
C                STORAGE FILE
C       ENIDE   TIME IN MILLISECS TO DELETE            I        13
C                SECONDARY STORAGE FILE
C       ENFIL   FOR FUTURE USE                         -        14 - 128
C
C***END OF COMMON BLOCK REFERENCE PHYCOM*****************************
```

Figure 6.8

corresponds to the byte or word size for most modern computers. SPPM parameters specify the SAU size in bits (e.g. 36 for the PDP-1Ø) and the number of characters that can be represented in a single SAU. A "Basic I/O Unit" is defined to be the smallest amount of secondary storage that is transferred for a single access; this term corresponds to the concept of sector or segment for a rotating device such as disk. Parameters for BIOU's define their size (in SAU's) and the maximum number that can be transferred with a single access. The latter concept refers to secondary storage allocation mechanisms that store files in non-contiguous groupings of BIOU's, sometimes termed "clusters". Another PHYCOM hardware parameter is a gross indicator of processor power. The NBS PDP-1Ø is used as the reference point. Currently, no distinction is made between processing and I/O capabilities; that is, a change in the power parameter is applied equally to all computer functions.

System load A single PHYCOM parameter represents the impact of other users on database performance. The simplicity of this treatment does not indicate a lack of understanding or appreciation for the complexity of resource allocation and scheduling problems, but rather reflects the high-level orientation of the SPPM modeling effort. Estimation of performance in a multi-programming environment is an interesting and challenging problem that has been widely addressed in the literature [SHER76, SVOB76, SALT7Ø, SCHW7Ø]. The objective of the SPPM modeling effort is to evaluate DBMS design concepts; hardware and operating system facilities are seen as the foundation upon which the DBMS is built.

I/O times The remaining PHYCOM parameters describe input/output capabilities by specifying milliseconds required for various direct access I/O functions. A secondary storage direct access read or write is viewed as being made up of three components.

* Access - arm movement and latency (rotational delay) for each reference sequence of a file.

* Transfer - actual transfer of data from main memory, through the channel, onto the device (or vice versa).

* Software Overhead - processing of DBMS, language, and operating system I/O software.

6.4.4 (SOFCOM) DBMS software. Parameters describing DBMS software appear in the common definition file SOFCOM that is reproduced in figure 6.9. A database management system can be viewed as a collection of functional components. The SPPM allows the user to define up to 100 DBMS functions and to optionally specify processing times and modification (improvement or degradation) factors for each.

6.4.5 (SPRCOM)Set processor. Common definition file SPRCOM contains parameters unique to the Positional Set Processor prototype. Set processor parameters appear in figure 6.10. Eventually, these parameters should be recast so that they apply to a broader class of database management system implementations; this has not yet been achieved, however.

6.5 Size Estimation Modeler

The SPPM size estimation facility is an analytic model that is applicable to a wide range of database management system designs using the relational logical view. The generality of the size modeler is dependent on the elementary file representation of secondary storage and its applicability to various DBMS storage structures. While this research has focused on a single DBMS design, the applicability of FILCOM parameters to various prototype and commercial systems has been considered. Many existing database management systems appear to be describable using SPPM parameters. The ability of the model to represent the complex and sometimes unique secondary storage structures employed by the Positional Set Processor prototype supports claims of power and flexibility. Model generality is enhanced by the ability to define optional parameter functions; a number of these user written procedures were used to describe PSP secondary storage structures.

The model calculates size estimates for source and stored databases. Stored database sizes are analyzed by secondary storage functions and by elementary files and their logical entries. Results are placed in common storage so that they can be accessed by other SPPM modules. The common definition file ZESCOM, which contains annotated size modeler derived variables, is reproduced in figure 6.11.

6.6 Response Estimation Modeler

The response estimation modeler is properly viewed as a modeling framework with utility functions necessary for a detailed modeling effort. Within this framework, algorithms for estimating response time for specific object DBMS designs can be developed using the SPPM provided utility

```
C*****************************************************
C
C       SOFCOM - COMMON BLOCK REFERENCE
C
C       PARAMETERS DESCRIBING THE DBMS SOFTWARE
C
C***********************************************************
C
C   - GLOBAL SOFTWARE PARAMETERS
C
        COMMON/SGSCOM/GSNFN,GSFIL(126)
C
        INTEGER GSNFN,GSFIL
C
C     PARAMETER       DESCRIPTION                      DEF    BUF SUB
C
C     _________   ___________________________          ___    _______
C
C       GSNFN     NO. OF DBMS FUNCTIONS DEFINED        I        1
C       GSFIL     FOR FUTURE USE                       -        2 - 128
C
C   - SOFTWARE FUCTIONS
C
        COMMON/SFNCOM/FNNAM(100),FNPRC(100),FNMOD(100),FNFIL(100,2)
     1                         FNSEC(12)
C
        INTEGER FNNAM,FNPRC,FNFIL,FNSEC
C
C     PARAMETER       DESCRIPTION                      DEF    BUF SUB
C
C     _________   ___________________________          ___    _______
C
C       FNNAM     NAME OF DBMS FUNCTION                A5       1 - 100
C       FNPRC     MILLISECS OF PROCESS TIME FOR        I        101-200
C                  EACH EXECUTION OF FUNCTION
C       FNMOD     MODIFICATION FACTOR FOR DBMS         F        201-300
C                  FUNCTION PROCESS TIME WHERE:
C                  TIME FOR I-TH FUNCTION =
C                   FNPRC(I)+FNMOD(I)*FNPRC(I)
C                  FNPRC(I)=NO SPECIFICATION => 0
C       FNFIL     FOR FUTURE USE                       -        301-500
C       FNSEC     FILL TO SECTOR BOUNDARY              -        501-512
C
C***END OF COMMON BLOCK REFERENCE SOFCOM**************************
C
```

Figure 6.9

```
C*************************************************************
C
C       SPRCOM - COMMON BLOCK REFERENCE
C
C       PARAMETERS DESCRIBING THE SET PROCESSOR DBMS
C
C*******************************************************************
C
        DOUBLE PRECISION KFILE
C
C   - BIT STRING PROCESSING PARAMETERS
C
        COMMON/SBSCOM/BSPKS,BSQLV,BSMPI,BSYNT,BSX1C,BSX2C,BSFIL(122)
C
        INTEGER BSPKS,BSQLV,BSMPI,BSFIL
C
C    PARAMETER      DESCRIPTION                         DEF    BUF SUB
C
C    _________   ___________________________           ___    _______
C
C       BSPKS    NO. OF BITS IN QUATREE PACKET          I        1
C       BSQLV    NO. OF QUATREE LEVELS                  I        2
C       BSMPI    MAX NO. OF POSITION ID'S FOR
C                 FILE STRUCTURE                        I        3
C
C.......(PARAMETERS FOR TRAVERSAL PROCESSING TIME ESTIMATION).......
C
C       BSYNT    Y INTERCEPT IN LINEAR EQUATION         F        4
C       BSX1C    COEF FOR X1 = CARDINALITY              F        5
C       BSX2C    COEF FOR X2 = RANGE                    F        6
C
C       BSFIL    FOR FUTURE USE                         -        7 - 128
C
C***END OF COMMON BLOCK REFERENCE SPRCOM****************************
C
```

Figure 6.1Ø

```
C******************************************************************
C
C       ZESCOM - COMMON REFERENCE
C
C       COMMON AREA FOR SIZE ESTIMATION COMPONENT OF PSPM
C
C*******************************************************************
C
C - GLOBAL SIZE ESTIMATION PARAMETERS
C
        COMMON/ZGZCOM/GZFIN,GZDAT(2),GZTIM,GZFIL(123)
C
        DOUBLE PRECISION GZFIN
        INTEGER GZDAT,GZTIM,GZFIL
C
C    PARAMETER      DESCRIPTION                    DEF    BUF SUB
C
C    _________  ____________________________       ___    _______
C
C       GZFIN   FILE NAME FOR SIZE ESTIMATION      (D)
C                REPORT (FFFFFF.SIZ)               A1Ø     1 - 2
C       GZDAT   DATE OF SIZE EST (DD-MM-YY)        2A5     3 - 4
C       GZTIM   TIME OF SIZE EST RUN (HH:MM)       A5      5
C       GZFIL   FOR FUTURE USE                     -       6 - 128
C
C  - SOURCE DATABASE SIZE ESTIMATION
C
        COMMON/ZSOCOM/SOTRL,SOTAT,SOTSZ,SOTIA,SOTUP,SOTRZ,
     1                  SOTAR,SOATN,SOTPR,SOFIL(119)
C
        INTEGER SOTRL,SOTAT,SOTIA,SOFIL,SOTAR,SOATN
C
C    PARAMETER      DESCRIPTION                    DEF    BUF SUB
C
C    _________  ____________________________       ___    _______
C
C       SOTRL   TOTAL NO. OF RELATIONS IN
C                SOURCE DB(INCL REPL REL)          I       1
C       SOTAT   TOTAL NO. OF ATTRIBUTES IN
C                SOURCE DB(INCL REPL ATT)          I       2
C       SOTSZ   TOTAL SIZE OF SOURCE DB IN
C                BITS FOR ALL RELS(W/REPL)         F       3
C       SOTIA   TOTAL NO. OF INDEXED ATT IN
C                ALL RELS                          I       4
C       SOTUP   TOTAL NO. OF TUPLES IN ALL
C                RELATIONS IN DB                   F       5
C       SOTRZ   TOTAL SIZE IN BITS OF ALL
C                NAMED RELS(W/O REPL)              F       6
C       SOTAR   TOTAL ATTRIBUTES INCLUDING
C                REPLICATED RELATIONS              I       7
C       SOATN   TOTAL INDEXED ATTRIBUTES
C                INCL REPLICATED RELATIONS         I       8
C       SOTPR   TOTAL TUPLES INCLUDING
C                REPLICATED RELATIONS              I       9
C       SOFIL   FOR FUTURE USE                     -       1Ø - 128
```

Figure 6.11a

```
C - STORED DB SIZE ESTIMATION

      COMMON/ZPSCOM/PSTID,PSTPR,PSTSR,PSTDE,PSTFO,PSEFZ,
     1                  PSTLE,PSTEF,PSTSZ,PSFIL(119)

      INTEGER PSTEF,PSFIL

C   PARAMETER       DESCRIPTION                      DEF    BUF SUB
C   _________    ___________________________        ___    _______

C      PSTID    TOTAL SIZE IN BITS OF DATA
C                INSTANCE STORAGE                    F        1
C      PSTPR    TOTAL SIZE IN BITS OF
C                PRIMARY RELATIONSHIPS               F        2
C      PSTSR    TOTAL SIZE IN BITS OF
C                SECONDARY RELATIONSHIPS             F        3
C      PSTDE    TOTAL SIZE IN BITS OF
C                DEFINITION                          F        4
C      PSTFO    TOTAL SIZE IN BITS OF
C                FILE OVERHEAD                       F        5
C      PSEFZ    SIZE IN BITS OF ALL NAMED
C                EF'S(W/Ø REPL)                      F        6
C      PSTLE    TOTAL NO. OF LE'S IN ALL
C                EF'S(INCL. REPL)                    F        7
C      PSTEF    TOTAL NO. OF EF'S IN DB
C                (INCL. REPL)                        I        8
C      PSTSZ    TOTAL SIZE IN BITS OF
C                STORED DB(ALL EF'S W/REPL)          F        9
C      PSFIL    FOR FUTURE USE                       -        1Ø-128
C
C - SIZES AND NUMBERS OF STORED ELEMENTARY FILES AND LOGICAL ENTRIES
C
      COMMON/ZSNCOM/SNEFO(1ØØ),SNFLS(1ØØ),SNFOH(1ØØ),SNLEO(1ØØ),
     1          SNLES(1ØØ),SNLEF(1ØØ),SNLOH(1ØØ),SNFIL(1ØØ,1),SNSEC(96)
C
C   PARAMETER       DESCRIPTION                      DEF    BUF SUB
C
C   _________    ___________________________        ___    _______
C
C      SNEFO    NO OF OCCURENCES FOR EF              F        1 - 1ØØ
C      SNFLS    TOTAL SIZE FOR ALL LE'S IN EF        F        1Ø1-2ØØ
C      SNFOH    OVERHEAD FOR EF                      F        2Ø1-3ØØ
C      SNLEO    NO OF OCCURENCES FOR LE              F        3Ø1-4ØØ
C      SNLES    SIZE FOR LE(EXCLUDING OVHD)          F        4Ø1-5ØØ
C      SNLEF    NO OF OCCURENCES FOR ALL LE'S        F        5Ø1-6ØØ
C                IN EF
C      SNLOH    OVERHEAD FOR LE                      F        6Ø1-7ØØ
C      SNFIL    FOR FUTURE USE                       -        7Ø1-8ØØ
C      SNSEC    FILL TO SECTOR BOUNDARY              -        8Ø1-896
C
C**END OF COMMON BLOCK REFERENCE ZESCOM*********************************
C
```

Figure 6.11b

functions. Figure 6.12 is a pictorial overview of the functional components of the response estimation modeler for the SPPM. A query in the form recognized by the PSP object DBMS is decoded and placed in a canonical form. A functional sequence selector determines the series of calls to set processor primitives and response model utilities required to estimate response time for the given query. Incremental estimates and activity indicators are posted to accumulators defined in common definition file RESCOM. Response time and I/O estimates for each query and for an entire query sequence are produced. Throughout the estimation process, references are made to parameters and derived size estimates in main memory rather than to an actual database. Monte Carlo processes are used whenever selection among alternatives can not be determined from user inputs and/or from parameter values. Response estimation modeler components are discussed in the following paragraphs.

6.6.1 Query. The SPPM accepts queries in the same format as those processed by the Positional Set Processor DBMS that is the object of this modeling effort. Like the PSP, the model utilizes the LANGPAK interactive language design and front-end parser for decoding queries. In fact, the SPPM decoder is made up of PSP programs with few if any modifications. The canonical form is recorded in parser arrays with complex boolean predicates converted to post-fix format.

6.6.2 Functional sequence selector. Given a query in canonical form, a DBMS must determine the sequence of activities required to generate a response. The sequence selector performs this function for the SPPM. Comprised of sometimes heavily modified programs from the Positional Set Processor object, it evaluates the query in relation to parameter descriptions of database contents and storage strategy and invokes a sequence of calls to utility procedures and to set processor primitives. Estimates of time and resource requirements for database functions representing events that would be performed by the PSP are recorded in the order in which they would be invoked. Replications of events that would be performed iteratively are handled by a pseudo process iteration indicator that is used as a multiplier when requirements are posted. The PSP functions necessary to answer a specific query are represented in the SPPM by a series of two types of events: calls to set processor primitives, and calls to response modeler utilities.

6.6.3 Set processor primitives. Set processor primitives are subroutines invoked when, in addition to estimating time and resource requirements, it is necessary to determine values for deriving subsequent processing steps. This is the case, for example, when the cardinality of a set resulting from a query selection must be determined to estimate the cost of

RESPONSE ESTIMATION MODELER OVERVIEW

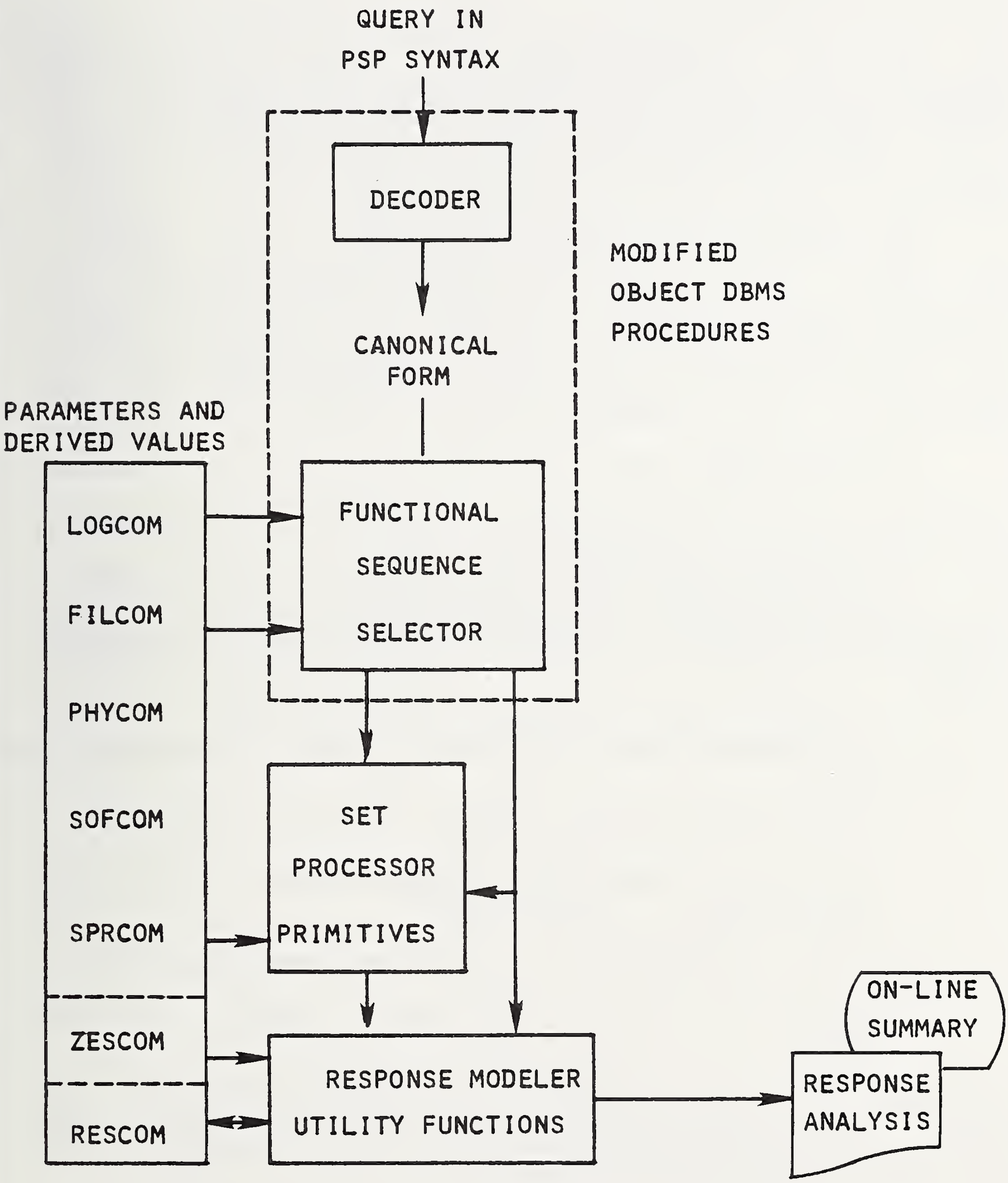


FIGURE 6.12

set operations that follow. Set processor primitives generally call SPPM utility functions to update response estimation accumulators. When there are no such processing requirements, time and resource estimates are posted by the appropriate response modeler utilities invoked directly by sequence selector programs.

For the initial SPPM implementation, twelve set processor primitives listed and described in figure 6.13 have been defined. This limited set of functions appears to be adequate for modeling PSP query response processing. Of course, additional experience applying the model to increasingly diverse designs may provide insight that will suggest additions to this list of primitives.

The primitive set traversal function TRVRS is used in two ways. It is, like other set processor primitives, invoked by sequence selector programs when traversal is a necessary step in the determination of an answer to a query. It is also used as a surrogate for estimating time and I/O requirements for many of the other primitive functions. For example, the cost of the operation

A UNION B ==> C

is approximated by the sum of the time and I/O requirements for traversing the three sets A, B and C.

<u>6.6.4 Response modeler utilities.</u> Response modeler utility functions are invoked by sequence selector and set processor primitive programs. Utilities perform four types of functions:

* retrieving times and/or posting model time accumulators ;
* maintaining a pseudo process iteration indicator;
* clearing and displaying response modeler result accumulators;
* estimating I/O resource and time requirements; and
* estimating traversal time and I/O resource requirements.

With the exception of traversal, all response utilities are general in that they operate within the context of the SPPM but are not specific to the Positional Set Processor prototype and its model representation.

SET PROCESSOR PRIMITIVES

FUNCTION NAME	DESCRIPTION
ALLOC	Allocate buffer for set representation
DALOC SSAVE SDEST	Release buffer: Save set representation on disk Destroy set representation in buffer
UNION	Union two sets
INTRS	Intersect two sets
XUNSD	Exclusive union two sets
RLCMP	Relative complement (set difference)
SCOPY	Reproduce set representation in other buffer
RGSTR	Find table entry and return reference; if not there, make table entries
SADD1	Add element to set
TRVRS	Enumerate (materialize) set members

Figure 6.13

Retrieve/post utilities A number of response utility functions are provided for retrieving parameter times for DBMS functions and for posting estimates to model accumulators. These functions are invoked by user written sequence selector programs and by set processor primitives.

Pseudo process iteration Utilities allow the user to increment and reset a pseudo process iteration indicator that is used as a multiplier when model estimates are posted to accumulators. These utilities warn the user of possible anomalies when nested iterations are encountered and check for consistency between indicator increments and decrements.

Accumulator initialization and display Model utilities are provided for initializing and generating on-line and hard copy reports from response modeler accumulators. On-line summary reports are generated and query level accumulators are cleared by invoking the appropriate utilities after processing each query. Other utilities generate detailed analyses for the entire response estimation session.

I/O estimation A generalized I/O estimation facility is used to predict and record I/O time and resource requirements. I/O requests are stated in terms of elementary file names appearing in FILCOM parameters, the starting SAU in the EF, and the numbers of smallest addressable units for each request and for the total amount of data to be transferred. The I/O estimator is patterned after the generalized direct access I/O facility in the PSP prototype; the conversion of I/O requests stated in terms of SAU's to specific BIOU transfers is done by the high level I/O interface. The I/O modeler is actually one level above the PSP facility in that it can model a sequence of logical I/O requests.

Traversal estimation Traversal estimation in the initial SPPM implementation is based on the bit-string set representations used in the PSP prototype. First, size of the bit string required to represent the set being traversed is estimated. Then, processing time is estimated based on parameters for packet size and cost to retrieve a packet. Finally, I/O costs are determined based on the size of the main memory buffer relative to the bit string set representation. Because of its specificity with respect to the object DBMS, the traversal estimation facility is the only response modeler utility that would have to be rewritten in order to model another set processor design strategy.

6.6.5 (RESCOM) Response modeler common. Accumulators and intermediate variables used by the response estimation modeler are defined in common definition file RESCOM. Figure 6.14 is a reproduction of this annotated FORTRAN common definition file.

```
C**********************************************************************
C
C       RESCOM - COMMON BLOCK REFERENCE
C
C       COMMON AREA FOR RESPONSE MODELER COMPONENT OF PSPM
C
C************************************************************************
C
C    - GLOBAL RESPONSE MODELER PARAMETERS
C
        COMMON/RGRCOM/GRNRL,GRFIN,GRDAT(2),GRTIM,
     1                 GRTRR,GRTWR,GRTOP,GRTCL,GRTDE,GRBTR,
     2                 GRACC,GRTRN,GROTI,GRTIO,
     3                 GRRRQ,GRWRQ,GROPQ,GRCLQ,GRDEQ,GRBTQ,
     4                 GRACQ,GRTNQ,GROTQ,GRTIQ,
     5                 GRTPQ,GRTPR,GRTIP,GRTOQ,GRTOH,
     6                 GRREQ,GRRES,GRPII,GRFIL(94)
C
        DOUBLE PRECISION GRFIN
        INTEGER GRNRL,GRDAT,GRTIM,GRTRR,GRTWR,GRTOP,GRTCL,GRTDE,GRBTR,
     1          GRRRQ,GRWRQ,GROPQ,GRCLQ,GRDEQ,GRBTQ,GRPII,GRFIL
C
C    PARAMETER       DESCRIPTION                     DEF    BUF SUB
C    _____ _____     ____________________________    ___    _______
C
C       GRNRL    NO. OF RELATION DEFS IN DB
C                 PARAMETER SET (INCL NEW RELS       I         1
C                CREATED DURING RESPONSE EST.)
C       GRFIN    FILE NAME FOR RESPONSE EST          (D)
C                 REPORT (FFFFFF.SIZ)                A1Ø       2 - 3
C       GRDAT    DATE OF RESPONSE ESTIMATION RUN
C                 (DD-MMM-YY)                        2A5       4 - 5
C       GRTIM    TIME OF RESPONSE ESTIMATION RUN
C                 (HH:MM)                            A6        6
C
C.......(TOTAL I/O ESTIMATES FOR SESSION, ALL E-FILES)..............
C
C       GRTRR    NO. OF READ REQUESTS                I         7
C       GRTWR    NO. OF WRITE REQUESTS               I         8
C       GRTOP    NO. OF FILE OPENS                   I         9
C       GRTCL    NO. OF FILE CLOSES                  I         1Ø
C       GRTDE    NO. OF FILE DELETES                 I         11
C       GRBTR    NO. OF BIOU'S TRANSFERRED           I         12
C       GRACC    ACCESS TIME                         F         13
C                 (ARM MOVEMENT + LATENCY)
C       GRTRN    TRANSFER TIME                       F         14
C       GROTI    OTHER I/O TIME                      F         15
C       GRTIO    TOTAL I/O TIME FOR SESSION          F         16
C
```

Figure 6.14a

```
C.......(TOTAL I/O ESTIMATES FOR QUERY, ALL E-FILES)................
C
C       GRRRQ   NO. OF READ REQUESTS                I       17
C       GRWRQ   NO. OF WRITE REQUESTS               I       18
C       GROPQ   NO. OF FILE OPENS                   I       19
C       GRCLQ   NO. OF FILE CLOSES                  I       2Ø
C       GRDEQ   NO. OF FILE DELETES                 I       21
C       GRBTQ   NO. OF BIOU'S TRANSFERRED           I       22
C       GRACQ   ACCESS TIME                         F       23
C                (ARM MOVEMENT + LATENCY)
C       GRTNQ   TRANSFER TIME                       F       24
C       GROTQ   OTHER I/O TIME                      F       25
C       GRTIQ   TOTAL I/O TIME FOR QUERY            F       26
C
C.......(TOTAL I/O AND PROCESSING ESTIMATES FOR ALL FUNCTIONS).......
C
C       GRTPQ   QUIESCENT SYSTEM PROCESSING         F       27
C                TIME FOR CURRENT QUERY
C       GRTPR   QUIESCENT SYSTEM PROCESSING         F       28
C                FOR SESSION
C       GRTIP   TOTAL I/O AND PROCESS TIME          F       29
C                FOR SESSION
C       GRTOQ   TOTAL SYSTEM OVERHEAD FOR QUERY F           3Ø
C       GRTOH   TOTAL SYSTEM OVHD FOR SESSION       F       31
C
C.......(TOTAL RESPONSE TIME ESTIMATES - I/O AND PROCESSING.......
C
C       GRREQ   RESPONSE TIME FOR CURRENT QUERY F           32
C       GRRES   RESPONSE TIME FOR SESSION           F       33
C
C.......(PSEUDO PROCESS ITERATION INDICATOR)....................
C
C       GRPII   PSEUDO PROCESS ITERATION IND        I       34
C       GRFIL   FOR FUTURE USE                      I       35-128
C
C   - WORKSPACE PARAMETERS
C
        COMMON/RWKCOM/WKRLB(1ØØ),WKFIL(28)
        INTEGER WKRLB,WKFIL
C
C       ----------------------------------------------------
C       PARAMETERS DESCRIBING PSPM REPRESENTATION OF SET-P
C         IN-CORE WORKSPACE BUFFERS
C
C       WKRLB(IRL)       = INDICATOR OF WHETHER RELATION IRL
C                          IN PARAMETER SET IS IN THE MAIN
C                          MEMORY BUFFERS FOR MODELING PURPOSES
C
C                        = 1 <==> IRL-TH RELATION IS IN WORKSPACE
C                          OTHERWISE, IRL-TH RELATION NOT IN WORKSPACE
C
C       WKFIL            FOR FUTURE USE
```

Figure 6.14b

```
C   - INTERNAL SET REPRESENTATION PARAMETERS
C
       COMMON/RSRCOM/SRNAM(100),SRCRD(100),SRRNG(100),SRLEN(100),
    1                SRUSB(100),SRNLE(100),SRSAU(100),SRCHG(100),
    2                SRFIL(100,3),SRLIS,SRSEC(51)
C
       INTEGER SRNAM,SRCRD,SRLEN,SRUSB,SRNLE,SRSAU,SRCHG,
    1          SRFIL,SRLIS,SRSEC
C
C   PARAMETER     DESCRIPTION                       DEF   BUF SUB
C
C   _________  ___________________________          ___   _______
C
C      SRNAM   INTERNAL SET(I.S.) NAME              A5      1 - 100
C      SRCRD   I.S. CARDINALITY                     I       101-200
C      SRRNG   RANGE OVER WHICH I.S. DEFINED        F       201-300
C               I.E., LARGEST INTEGER
C      SRLEN   SEC STORAGE LOGICAL ENTRY TYPE       A5      301-400
C               'SC'==> MODE = SCRATCH
C      SRUSB   USER I.S. BUFFER SIZE IN SAU'S       I       401-500
C      SRNLE   ORDINAL FOR LE W/IN ALL LE'S         I       501-600
C               IN EF
C      SRSAU   STARTING SAU OF PORTION OF LE        I       601-700
C               IN USER BUFFER
C      SRCHG   IND OF WHETHER PORTION OF SET                701-800
C               IN BUFFER HAS BEEN CHANGED
C              = 0 <==> NO MODIFICATION
C      SRFIL   FOR FUTURE USE                       -       801-1100
C
C.......(GLOBAL INDEX TO LAST INTERNAL SET DEFINED)...........
C
C      SRLIS   LAST INTERNAL SET DEFINED            I       1101
C
C      SRSEC   FILL TO SECTOR BOUNDARY              I       1102-1152
C
C   - SET REPRESENTATIN PARAMETER SUBSCRIPTS
C
       COMMON/RSSCOM/ISSNAM,ISSCRD,ISSRNG,ISSLEN,ISSNLE,ISSFIL(123)
C
C   PARAMETER     DESCRIPTION                       DEF   BUF SUB
C
C   _________  ___________________________          ___   _______
C
C      ISSNAM  SUBSCRIPT FOR SRNAM                  I       1
C      ISSCRD  SUBSCRIPT FOR SRCRD                  I       2
C      ISSRNG  SUBSCRIPT FOR SRRNG                  I       3
C      ISSLEN  SUBSCRIPT FOR SRLEN                  I       4
C      ISSNLE  SUBSCRIPT FOR SRNLE                  I       5
C
C      ISSFIL  FOR FUTURE USE                       I       6-128
```

Figure 6.14c

```
C   - PROCESSOR UTILIZATION ESTIMATES BY SOFTWARE FUNCTION
C
      COMMON/RPRCOM/PRNEX(100),PRTPR(100),PRTIO(100),PRTIP(100),
     1          PRFIL(100,3),PRONX,PROPR,PROIO,PROIP,PRSEC(64)
      INTEGER PRNEX,PRONX,PRFIL,PRSEC
C
C    PARAMETER       DESCRIPTION                        DEF    BUF SUB
C
C    _________   ___________________________         ___    _______
C
C       PRNEX    NO. OF EXECUTIONS FOR DBMS          I        1 - 100
C                 FUNCTION
C       PRTPR    TOTAL QUIESCENT SYSTEM PRO-         F        101-200
C                 CESSING TIME FOR DBMS FUNC
C       PRTIO    TOTAL I/O TIME FOR FUNCTION         F        201-300
C       PRTIP    TOTAL I/O AND PROCESS TIME          F        301-400
C       PRFIL    FOR FUTURE USE                      I        401-700
C
C.......(FUNCTION 'OTHER' ACCUMULATORS)......................
C
C       PRONX    NO OF EXECTUIONS, OTHER             I        701
C       PROPR    QUIESCENT SYSTEM PROC TIME          F        702
C       PROIO    I/O TIME, OTHER                     F        703
C       PROIP    TOTAL I/O AND PROCESS TIME          F        704
C
C...............................................................
C
C       PRSEC    FILL TO SECTOR BOUNDARY             I        705-768
C
C   - I/O STATISTICS BY ELEMENTARY FILE AND EF BUFFER SIZE
C
      COMMON/RIECOM/IENRR(100),IENWR(100),IEBTR(100),IEOTH(100),
     1                  IETIM(100),IEBUF(100),IEFIL(100,2),IESEC(96
      REAL IETIM
C
C    PARAMETER       DESCRIPTION                        DEF    BUF SUB
C
C    _________   ___________________________         ___    _______
C
C       IENRR    NO. OF READ REQUESTS                I        1 - 100
C       IENWR    NO. OF WRITE REQUESTS               I        101-200
C       IEBTR    NO. OF BIOU'S TRANSFERRED           I        201-300
C       IEOTH    NO. OF OTHER I/O REQUESTS           I        301-400
C                (INCL. OPEN, CLOSE, DELETE)
C       IETIM    TOTAL I/O TIME                      F        401-500
C       IEBUF    I/O SOFTWARE BUFFER CONTENTS:       I        501-600
C                    = 0 <==> NOT OPEN
C                    < 0 <==> FILE OPEN BUT BUFFER
C                             NOT FILLED FROM FILE
C                             (I.E. NO READ OCCURRED)
C                    > 0 <==> STARTING SAU ON FILE
C                             FOR BUFFER CONTENTS OF
C                             OPEN FILE
C       IEFIL    FOR FUTURE USE                      -        601-800
C       IESEC    FILL TO SECTOR BOUNDARY             -        801-896
```

Figure 6.14d

```
C  - RESPONSE MODELER EF AND LE INDICES
C
      COMMON/RSUCOM/IDMAIN,IALIAS,IELMNT,IETNDX,ITXTBL,ISNDXM,
     1                ISWORK,ISINVS,ISRAVI,ITEMEF,IFIL(2Ø),
     2                IATDEF,IREFEF,IATALI,IRLALI,ISETPI,IATPID,
     3                IATOM,ITUPLE,IRELAT,IEHASH,ISPNTR,IUNVRS,
     4                IAVSET,IRAVAL,IRAPTR,IRLBST,ITUBST,ITEMLE,
     5                IFIL2(22)
C
C    PARAMETER      DESCRIPTION                     DEF   BUF SUB
C
C    _________  ___________________________         ___   _______
C
C       IXXXXX  INDEX FOR EF XXXXX = DMAIN           I       1 - 1Ø
C                                    ALIAS
C                                    ELMNT
C                                    ETNDX
C                                    TXTBL
C                                    SNDXM
C                                    SWORK
C                                    SINVS
C                                    SRAVI
C                                    TEMEF
C
C       IFIL    FOR FUTURE USE                       I       11-3Ø
C
C       IYYYYY  INDEX FOR LE YYYYY = ATDEF           I       31-48
C                                    REDEF
C                                    ATALI
C                                    RLALI
C                                    SETPI
C                                    ATPID
C                                    ATOM
C                                    TUPLE
C                                    RELAT
C                                    EHASH
C                                    SPNTR
C                                    UNVRS
C                                    AVSET
C                                    RAVAL
C                                    RAPTR
C                                    RLBST
C                                    TUBST
C                                    TEMLE
C
C       IFIL2   FOR FUTURE USE                       I       49-6Ø
C
C***END COMMON BLOCK REFERENCE RESCOM*****************************
```

Figure 6.14e

6.6.6 Monte Carlo processors. Throughout the response estimation process, Monte Carlo processes are used for estimating specific LE occurrences and set cardinalities that can not be determined from user inputs and/or parameter values. The uniform distribution is used with the PDP-10 FORTRAN-10 random number generator.

6.7 SPPM Component Interaction

Set processor performance model components interact through parameters and derived values in common storage. Size modeler derived values are the basis for response modeler estimates; that is, response modeler I/O estimation routines reference elementary file and logical entry occurrence frequencies and sizes determined by the size modeler and stored in common ZESCOM.

An analysis of the approximately 215 FORTRAN subroutines that comprise the SPPM shows the following distribution.

SPPM COMPONENT		% OF ALL PROGRAMS
Model driver and SPPM utility modules		37%
Size modeler		23
Response modeler:		
General framework	16%	
PSP specific	10	26

Model utility subroutines		14
TOTAL - ENTIRE SPPM		100%

Thus, only 10% of all model subroutines are specific to the object DBMS. The other 90% would be appropriate to modeling other DBMS designs.

The dependency of the response modeler on existing size model estimates is reflected in the following scenario for applying the SPPM to an existing prototype with measurement capability.

1. Prepare FILCOM parameters describing secondary storage structures for the object DBMS.

2. Prepare PHYCOM and (for the PSP only) SPRCOM parameters describing the hardware/software environment and DBMS design specific characteristics.

3. Prepare one or more LOGCOM parameter sets describing logical databases and their contents.

4. Use the SPPM size modeler to estimate database size.

5. Review and validate size estimates. Iterate on steps 1 through 4 until satisfied with the model representation of the DBMS.

6. Use the measurement system to provide insight into operation of the DBMS prototype and to prepare a preliminary SOFCOM parameter list.

7. Prepare DBMS specific decoder and sequence selector procedures referencing defined and derived characteristics for specific elementary files and logical entries defined in step 1 and stored in common blocks FILCOM and ZESCOM. Revise SOFCOM parameter list as required.

8. Use the measurement system to determine parameter values for SOFCOM parameters as revised.

9. Use the response modeler to estimate response time and I/O resource requirements.

10. Review and validate response estimates. Iterate on steps 6 through 8 until desired accuracy is achieved.

11. Perturb parameters and query complexity to determine performance beyond range of object DBMS prototype capability and to study the impact on performance of potential changes in DBMS design and environment.

The next chapter describes in greater detail SPPM size and response estimation outputs and derives mathematical relationships using the parameters defined above.

7. PERFORMANCE MODEL - MATHEMATICAL SUMMARY

7.1 Introduction

This chapter presents a summary of the mathematical relationships that are embedded in the SPPM programs. Like the prototype DBMS that is the object of the modeling effort, the Set Processor Performance Model is coded entirely in (relatively) standard FORTRAN for the DEC/PDP-1Ø computer. Approximately 125 subroutines comprise the size estimation and response modelers. Another 8Ø subroutines perform the driver, help, display, change and I/O functions described in the previous chapter. Because of the time and difficulty associated with reading even well structured and documented programs such as these, essential mathematical relationships have been extracted, recorded in a pseudo-FORTRAN notation, and briefly annotated.

7.1.1 Model documentation: form and content. The author believes that a concise statement of mathematical relationships is a necessary component of model documentation. Complex computerized models such as the SPPM have both mathematical and software characteristics. While the questions of form and content for documentation of software systems are addressed by a substantial body of literature and practical guidelines [KRAS77, BENJ71, NBS76], procedures for documenting large computerized models are not as well developed [GASS79]. Thus, the format used in the following paragraphs reflects the SPPM source language (FORTRAN) as well as the author's ideas about the information that should be included in a summary of mathematical relationships. It should be noted, however, that material like that contained in this chapter is necessary but not sufficient for describing model software. Other required supporting documents include system and program logic flow charts, source program listings, parameter (common block) listings, operating instructions, and narrative descriptions for the overall model and its components [GASS79].

7.1.2 Parameters, indices and index functions. The mathematical relationships in the following paragraphs are stated in terms of the parameters defined in the previous chapter. In general, these parameters are used for expressing relationships in this chapter without further definition. An exception is made when parameters appear directly on output reports; these parameters are defined again for completeness and ease of understanding.

Model parameters are of two types: single valued (scalars) and multi-valued (arrays). While the former can be referenced directly, the latter must be referenced using a clarifying subscript. Multi-valued parameters are associated with the seven entity classes listed below.

* Relations
* Attributes
* Domains
* Elementary Files
* Logical Entities
* Database Management Functions
* Set Representations

Special index variables, corresponding to these seven entity classes, are used consistently throughout for clarifying references to multi-valued parameters. In general, model references are always by name; that is, subroutine arguments are entity (relation, attribute, domain, elementary file, etc.) names rather than index values. Consequently, index functions are used to translate name references into their corresponding subscript values. Figure 7.1 lists the seven index variables and their associated index functions.

Residing in common (shared) main memory, these indices can be accessed and modified by all SPPM subroutines. Parameter indices serve as "currency" indicators; that is, they reflect the specific entity parameters that are currently being considered by the model. An index value is properly viewed as an ordinal within all defined parameters for an entity class. For brevity, however, references are made to "relation IRL", and "logical entry ILE", etc. rather than to the more precise "IRL-th relation defined in the parameter set", or "ILE-th logical entry defined in the parameter set", etc. throughout this document.

<u>7.1.3 Chapter overview.</u> The remainder of this chapter summarizes SPPM mathematical relationships using an annotated pseudo-FORTRAN notation under four major headings. First, model utility routines are listed. The next two sections describe size estimation model relationships and present the mathematics for the response modeler respectively. Both the size and response models are described in terms of variables appearing on their respective output reports. Finally, bit string size estimation and user defined functions for

SPPM INDEX VARIABLES AND FUNCTIONS

INDEX	DESCRIPTION	FUNCTION
IRL	Relation index	IRLSUB(relation name)
IAT	Attribute index	IATSUB(attribute name)
IDM	Domain index	IDMSUB(domain name)
IEF	Elementary file index	IEFSUB(elementary file name)
ILE	Logical entry index	ILESUB(logical entry name)
IFN	DBMS function index	IFNSUB(DBMS function name)
ISR	Set representation index	ISRSUB(set representation name)

Figure 7.1

determining parameter values are discussed. Mathematical foundations for (two) major classes of functions written for modeling the Positional Set Processor Prototype are presented. A list of all currently defined parameter functions also appears.

Despite their apparent complexity and completeness, the mathematical definitions that follow do not fully describe the SPPM size and response estimation facilities. While the model is encoded in a procedural language, this chapter presents model relationships using a non-procedural mathematical notation. Artificial indicator variables are defined to allow representation of conditional statements embedded in iteration procedures. Other, more complex procedural interactions must be discovered through review of source code for SPPM programs, however.

7.2 Model Utility Routines

A number of utility functions are used by the SPPM. These routines are defined here so that they can be used for representing model mathematical relationships. The functions listed below are mathematical primitives that are used throughout the size and response modelers; their meaning is not tied to SPPM procedures or parameters.

INTUP(VALUE)	=	the smallest integer greater than or equal to real VALUE.
INTDN(VALUE)	=	the largest integer less than or equal to real VALUE.
IGCD(N1,N2)	=	the greatest common divisor for the pair of integers, N1 and N2; this function uses the Euclidean algorithm.
MAXINT(N1,N2)	=	the largest of two integers, N1 and N2.
	=	N2 <==> N2 > N1 N1 otherwise
MININT(N1,N2)	=	the smallest of two integers, N1 and N2.
	=	N2 <==> N2 < N1 N1 otherwise

7.3 Database Size Estimation

The size estimation model produces three hardcopy detailed analysis reports and three on-line summaries derived from the hardcopy output. One detailed report and one summary display contain size estimates for source databases. Other outputs are concerned with database sizes after they have been loaded by the DBMS that is the object of the modeling effort. Source database size estimates provide benchmarks for evaluating the secondary storage utilization efficiency for the database management design being studied. For instance, one interesting measure of a DBMS design is its database explosion factor; that is, how much larger is a stored database than the source from which it was loaded? The following paragraphs consider mathematical relationships for estimating sizes of source and stored databases.

7.3.1 Source database size estimation. Figure 7.2 presents copies of the source database on-line summary and detailed analysis reports with variable names inserted in brackets under data entries. These variables are defined below.

Magnitudes for each relation occurrence First magnitudes for each occurrence of relations defined for the database are dervied. Given the following indicator variables:

INDRAT(IRL,IAT) = indicator of whether relation IRL contains attribute IAT

= Ø <==> RAINM(IRL,IAT) = Ø
1 Otherwise

IATRPL(IRL,IAT) = number of replications of attribute IAT in relation IRL

= |RAINM(IRL,IAT)|

INDXED(IRL,IAT) = indicator of whether attribute IAT is indexed in relation IRL

= Ø <==> RAINM(IRL,IAT) ≥ Ø
1 <==> RAINM(IRL,IAT) ≤ Ø

we count named, total and indexed attributes for each occurrence of relation IRL.

SOURCE DATABASE SUMMARY AND DETAILED ANALYSIS REPORTS

SOURCE DATABASE

ENTITIES	DEFINED	TOTAL DB
RELATIONS	4 [DBNRL]	4 [SOTRL]
ATTRIBUTES	11 [SOTAT]	11 [SOTAR]
TUPLES	41.000 [SOTUP]	41.000 [SOTPR]
SIZE(BITS)	18432. [SOTRZ]	18432. [SOTSZ]

SOURCE DATABASE SIZE ANALYSIS

RELNAM	ALL DEFINED RELATIONS					TOTAL DATABASE					
	ATTRIBUTES			TUPLES	REL SIZE	REL	ATTRIBUTES		TUPLES	DATABASE	TOTAL
	NAMED	TOTAL	INDXD		(BITS)	REPL	TOTAL	INDXED		TOTAL SIZE	BIOUS
[RLNAM]	[INMATR]	[ITOATR]	[INDATR]	[RLCRD]	[RELSIZ]	[RLRPL]	[ITARPL]	[ITANDX]	[TUPTOT]	[TRLSIZ]	[IRLBIU]
EMP	4	4	2	10.000	8280.0	1	4	2	10.000	8280.0	2
SALES	2	2	0	12.000	1296.0	1	2	0	12.000	1296.0	1
SUPLY	2	2	2	10.000	7560.0	1	2	2	10.000	7560.0	2
TYPE	3	3	1	9.0000	1296.0	1	3	1	9.0000	1296.0	1
TOTAL	11 [INMATD]	11 [SOTAT]	5 [SOTIA]	41.000 [SOTUP]	18432. [SOTRZ]	4 [SOTRL]	11 [SOTAR]	5 [SOATN]	41.000 [SOTPR]	18432. [SOTSZ]	6 [ISTBIU]

Figure 7.2

INMATR(IRL) = number of named attributes for relation IRL in source DB; does not include either attribute or relation replications.

$$= \sum_{IAT=1}^{DBNAT} INDRAT(IRL,IAT)$$

ITOATR(IRL) = total number of attributes for relation IRL in source DB; includes attribute replications but does not include relation replications.

$$= \sum_{IAT=1}^{DBNAT} IATRPL(IRL,IAT)$$

INDATR(IRL) = total number of indexed attributes for relation IRL in source DB; includes attribute replications but does not include relation replications.

$$= \sum_{IAT=1}^{DBNAT} INDXED(IRL,IAT)*IATRPL(IRL,IAT)$$

Then, the number of tuples and total size for each occurrence of relation IRL are determined from database parameters and the above derivations.

RLCRD(IRL) = number of tuples in relation IRL in source (by definition); does not include relation replications. (Attribute replications do not impact tuple count.)

RELSIZ(IRL) = total source DB size in bits for relation IRL; includes attribute replications but does not include relation replications.

$$= \sum_{IAT=1}^{DBNAT} IATRPL(IRL,IAT)*DMSCRD$$

where: DMSCRD=DMAVS(IDM)*RLCRD(IRL)
IDM=IDMSUB(ATDOM(IAT))

<u>Magnitudes for all relation replications</u> Given the number of relation replications:

RLRPL(IRL) = number of replications for relation IRL (by definition)

we calculate totals for all replications of relation IRL.

ITARPL(IRL) = total number of attributes for relation IRL in source DB including both attribute and relation replications.

= ITOATR(IRL) * RLRPL(IRL)

ITANDX(IRL) = total number of indexed attributes for relation IRL in source DB including both attribute and relation replications.

= INDATR(IRL) * RLRPL(IRL)

TUPTOT(IRL) = total number of tuples for relation IRL in source DB including relation replications (attribute replications do not impact tuple count).

= RLCRD(IRL) * RLRPL(IRL)

TRLSIZ(IRL) = total size in bits for relation IRL in source DB including both attribute and relation replications.

= RELSIZ(IRL) * RLRPL(IRL)

IRLBIU(IRL) = total size in Basic I/O Units (BIOU's) for relation IRL in source database including both attribute and relation replications.

= INTUP(TRLSIZ(IRL)/(ENBIU*ENSAU))

Totals for all relations Totals for the source database size analysis report are derived by summing the above results for specific relations across all relations defined for the database.

INMATD = total number of named attributes for all relations in source DB; does not include either attribute or relation replications. (N.B. a single named attribute may be contained in multiple relations.)

$$= \sum_{IRL=1}^{DBNRL} INMATR(IRL)$$

$$\geq DBNAT$$

SOTAT = total number of attributes for all relations in source DB; includes attribute replications but does not include relation replications.

$$= \sum_{IRL=1}^{DBNRL} ITOATR(IRL)$$

SOTIA = total number of indexed attributes for all relations in source DB; includes attribute replications but does not include relation replications.

$$= \sum_{IRL=1}^{DBNRL} INDATR(IRL)$$

SOTUP = total number of tuples for all relations in source DB; does not include relation replications. (Attribute replications do not impact tuple count.)

$$= \sum_{IRL=1}^{DBNRL} RLCRD(IRL)$$

SOTRZ = total source DB size in bits for all relations; includes attribute replications but does not include relation replications.

$$= \sum_{IRL=1}^{DBNRL} RELSIZ(IRL)$$

SOTRL = total number of relations in source DB including relation replications.

$$= \sum_{IRL=1}^{DBNRL} RLRPL(IRL)$$

SOTAR = total number of attributes for all relations in source DB including both attribute and relation replications.

$$= \sum_{IRL=1}^{DBNRL} ITARPL(IRL)$$

SOATN = total number of indexed attributes for all relations in source DB including both attribute and relation replications.

$$= \sum_{IRL=1}^{DBNRL} ITANDX(IRL)$$

SOTPR = total number of tuples for all relations in source DB including relation replications (attribute replications do not impact tuple count).

$$= \sum_{IRL=1}^{DBNRL} TUPTOT(IRL)$$

SOTSZ = total size in bits for source DB including both attribute and relation replications.

$$= \sum_{IRL=1}^{DBNRL} TRLSIZ(IRL)$$

ISTBIU = total size in Basic I/O Units (BIOU's) for source DB including both attribute and relation replications.

$$= \sum_{IRL=1}^{DBNRL} IRLBIU(IRL)$$

7.3.2 Stored database size estimation. Two on-line summaries and two hard copy analysis reports are produced by the stored database size estimation module. Figure 7.3 contains copies of these SPPM outputs with variable names inserted in brackets under data entries. The stored database size estimation process requires that occurrence frequencies and size estimates be determined for all logical entries. Logical entry statistics are accumulated by elementary file (EF) and by secondary storage functions. To estimate storage

STORED DATABASE SUMMARY REPORTS

STORED DATABASE FILES

ENTITIES	DEFINED	TOTAL DB
ELEM FILES	10 [GFNEF]	16 [PSTEF]
LOG ENTRIES	19 [GFNLE]	369.00 [PSTLE]
SIZE(BITS)	.34711E+06 [PSEFZ]	.48078E+06 [PSTSZ]

STORAGE UTILIZATION

STORAGE FUNCTION	SIZE(BITS)
PRIMARY RELATIONSHIPS	43487. [PSTPR]
SECONDARY RELATIONSHIPS	22026. [PSTSR]
DEFINITION	23040. [PSTDE]
DATA INSTANCES	28800. [PSTID]
FILE OVERHEAD	.36343E+06 [PSTFO]
TOTAL STORED DATABASE	.48078E+06 [PSTSZ]

Figure 7.3a

STORED DATABASE DETAILED ANALYSIS REPORTS

STORED DATABASE SIZE - ELEMENTARY FILE ANALYSIS

E-FILE NO	NAME [EFNAM]	NO LE TYPES [EFLET]	NUMBER LE'S [SNLEF]	TOTAL EF SIZE [EFSIZE]	EF REPL [SNEFO]	TOTAL ALL EF [TEFSIZ]	.........STORED DATABASE SIZE BY STORAGE FUNCTION......... PRIM REL [TSZPRF]	SECN REL [TSZSRF]	DEFN [TSZDEF]	INST DATA [TSZDIF]	OVERHEAD [SNFOH]	TOTAL BIOUS [IBIOUS]
1	DMAIN	2	15.000	8640.0	1	8640.0			8640.			2
2	ALIAS	2	20.000	14580.	1	14580.			.1440E+05		180.0	4
3	ELMNT	5	94.000	67860.	1	67860.	.3888E+05			.2880E+05	180.0	15
4	ETNDX	1	94.000	.11088E+06	1	.11088E+06		3384.			.1075E+06	25
5	TXTBL	2	45.000	37187.	1	37187.	4607.				.3258E+05	9
6	SNDXM	1	3.0000	36864.	1	36864.		.1382E+05			.2304E+05	8
7	SWORK	2	4.0000	13860.	1	13860.		354.0			.1351E+05	4
8	SINVS	1	31.000	47652.	3	.14296E+06		2232.			.1407E+06	33
9	SRAVI	2	62.000	9590.4	5	47952.		2232.			.4572E+05	15
10	TEMEF	1	1.0000	.00000E+00	1	.00000E+00						0
	TOTALS	19 [INLETD]	369.00 [PSTLE]	.34711E+06 [PSEFZ]	16 [PSTEF]	.48078E+06 [PSTSZ]	43487. [PSTPR]	22026. [PSTSR]	23040. [PSTDE]	28800. [PSTID]	.36343E+06 [PSTFO]	115 [ITBIOU]

STORED DATABASE SIZE - LOGICAL ENTRY ANALYSIS

NO [ILE]	NAME [LENAM]	EF REF [LEFRF]	SIZE(BITS) [SNLES]	STOFNC [LEFUN]	LE OVHD [SNLOH]	BITS/LE [TOTBIT]	LE OCCUR [SNELO]	LE FUNC [TLEFNC]	LE OVHD [TLEOHD]	TOTAL LE [TLESIZ]
1	ATDEF	DMAIN	576.00	DEF		576.00	11.000	6336.0		6336.0
2	REDEF	DMAIN	576.00	DEF		576.00	4.0000	2304.0		2304.0
3	ATALI	ALIAS	720.00	DEF		720.00	16.000	11520.		11520.
4	RLALI	ALIAS	720.00	DEF		720.00	4.0000	2880.0		2880.0
5	SETPI	ELMNT	720.00	PRI		720.00	1.0000	720.00		720.00
6	ATPID	ELMNT	720.00	PRI		720.00	8.0000	5760.0		5760.0
7	ATOM	ELMNT	720.00	INS		720.00	40.000	28800.		28800.
8	TUPLE	ELMNT	720.00	PRI		720.00	41.000	29520.		29520.
9	RELAT	ELMNT	720.00	PRI		720.00	4.0000	2880.0		2880.0
10	EHASH	ETNDX	36.000	SEC		36.000	94.000	3384.0		3384.0
11	SPNTR	SNDXM	4608.0	SEC	36.00	4644.0	3.0000	13824.	108.0	13932.
12	UNVRS	SWORK	118.00	SEC	4490.	4608.0	3.0000	354.00	.1347E+05	13824.
13	WRKLE	SWORK	.00000E+00	TMP		.00000E+00	1.0000	.00000E+00		.00000E+00
14	AVSET	SINVS	72.000	SEC	4536.	4608.0	31.000	2232.0	.1406E+06	.14285E+06
15	RAVAL	SRAVI	36.000	SEC		36.000	31.000	1116.0		1116.0
16	RAPTR	SRAVI	36.000	SEC		36.000	31.000	1116.0		1116.0
17	RLBST	TXTBL	219.00	PRI	720.0	939.00	4.0000	876.00	2880.	3756.0
18	TUBST	TXTBL	91.000	PRI	720.0	811.00	41.000	3731.0	.2952E+05	33251.
19	TEMLE	TEMEF	.00000E+00	OVH		.00000E+00	1.0000	.00000E+00		.00000E+00
	TOTALS						369.00 [TOTLEO]	.11735E+06 [TOTFNC]	.18659E+06 [TOTOHD]	.30395E+06 [TOTSIZ]

Figure 7.3b

required for logical entries, elementary file size calculations consider EF overhead, number of EF occurrences, and fixed length elementary files. Derivations for output variables, first for logical entries and then for elementary files, appear below.

Logical entry analysis For each logical entry type, the following parameters for the LE number, name, elementary file reference and storage function appear on the output report.

ILE	=	Index for logical entry type (by definition).
LENAM(ILE)	=	name for logical entry type ILE (by definition).
LEFRF(ILE)	=	elementary file reference for logical entry type ILE (by definition).
LEFUN(ILE)	=	secondary storage function for logical entry type ILE (by definition).

Size, overhead, and number of occurrences for each logical entry type, and overhead for each logical entry in an elementary file are obtained from FORTRAN functions defined over FILCOM parameters. These functions, described further in the "Parameter Functions" section at the end of this chapter, provide mechanisms for invoking intrinsic and optional parameter functions.

SNLES(ILE)	=	size in bits for each occurrence of logical entry type ILE; does not include overhead for LE, for EF logical entries, or for EF.
	=	LESIZE(ILE); function defined over parameter LESIZ(ILE).
LEOVHD(ILE)	=	overhead in bits for each occurrence of logical entry type ILE; does not include overhead for EF logical entries or for EF; function defined over parameter LEOHD(ILE).
SNLEO(ILE)	=	number of occurrences for logical entry type ILE for all elementary files.
	=	LEOCCR(ILE); function defined over parameters LENOC(ILE), LERFO(ILE) and LERFQ(ILE).

EFLEOH(ILE)	=	overhead in bits for each occurrence of logical entry type ILE in its elementary file; does not include LE or EF overhead.
	=	IEFEOH(IEF,ILE); function defined over parameter EFEOH(IEF).
		where: IEF = IEFSUB(LEFRF(ILE))

Using the above, overhead and total LE size for each logical entry occurrence are calculated.

SNLOH(ILE)	=	overhead for each occurrence of logical entry type ILE; includes LE and EF logical entry overhead, but does not include overhead for EF.
	=	LEOVHD(ILE) + EFLEOH(ILE)
TOTBIT(ILE)	=	total size in bits for each occurrence of logical entry type ILE; includes LE and EF logical entry overhead, but does not include EF overhead.
	=	SNLES(ILE) + SNLOH(ILE)

To determine totals for all occurrences of LE types, single occurrence sizes are extended by LE occurrence frequencies.

TLEFNC(ILE)	=	total size in bits for functional portions of all occurrences of logical entry type ILE; does not include any overhead.
	=	SNLES(ILE) + SNLEO(ILE)
TLEOHD(ILE)	=	total size in bits for overhead portions of all occurrences of logical entry type ILE; includes LE and EF logical entry overhead, but does not include overhead for EF.
	=	SNLOH(ILE) * SNLEO(ILE)
TLESIZ(ILE)	=	total size in bits for all occurrences of logical entry type ILE; includes LE and EF logical entry overhead, but does not include EF overhead.
	=	TOTBIT(ILE) * SNLEO(ILE)

Finally, totals for all logical entries in the database are determined by summing the results derived above for specific LE types across all LE's.

TOTLEO = total number of occurrences for all logical entry types.

$$= \sum_{ILE=1}^{GFNLE} SNLEO(ILE)$$

TOTFNC = total size in bits for functional portions of all occurrences of all logical entry types; does not include any overhead.

$$= \sum_{ILE=1}^{GFNLE} TLEFNC(ILE)$$

TOTOHD = total size in bits for overhead portions of all occurrences of all logical entry types; includes LE and EF logical entry overhead, but does not include overhead for EF's.

$$= \sum_{ILE=1}^{GFNLE} TLEOHD(ILE)$$

TOTSIZ = total size in bits for all occurrences of all logical entry types; includes LE and EF logical entry overhead, but does not include overhead for EF's.

$$= \sum_{ILE=1}^{GFNLE} TLESIZ(ILE)$$

<u>Elementary file analysis</u> The SPPM Elementary File Analysis output report contains line item entries for each elementary file defined in the parameter set and totals for the entire database. Many of the results derived during the analysis

of logical entries are used in estimating elementary file magnitudes; these variables are not redefined in this section.

For each defined elementary file, the parameters listed below for the EF number, EF name and number of logical entry types contained in the elementary file appear on the output report.

IEF = index for defined elementary file (by definition).

EFNAM(IEF) = name for elementary file IEF (by definition).

EFLET(IEF) = number of logical entry types appearing in elementary file IEF (by definition)

We define the following indicator variables to facilitate the calculation of total EF size and the allocation of storage estimates to secondary storage functions.

INDFLE(IEF,ILE) = indicator of whether elementary file IEF contains logical entry type ILE.

= 1 <==> LEFRF(ILE) = EFNAM(IEF)
Ø Otherwise

INDPRI(ILE) = indicator of whether logical entry type ILE represents primary relationships.

= 1 <==> LEFUN(ILE) = "P"
Ø Otherwise

INDSEC(ILE) = indicator of whether logical entry type ILE represents secondary relationships.

= 1 <==> LEFUN(ILE) = "S"
Ø Otherwise

INDEFN(ILE) = indicator of whether logical entry type ILE represents definition.

= 1 <==> LEFUN(ILE) = "D"
Ø Otherwise

INDATA(ILE) = indicator of whether logical entry type ILE represents data instances.

= 1 <==> LEFUN(ILE) = "I"
Ø Otherwise

Allocation of non-overhead portions of logical entries to elementary files and to secondary storage functions is performed in the following manner.

TSZPRF(IEF) = total bits for representing primary relationships in each occurrence of elementary file IEF.

$$= \sum_{ILE=1}^{GFNLE} INDFLE(IEF,ILE)*PRIFNC(ILE)$$

where: PRIFNC(ILE)=INDPRI(ILE)* TLEFNC(ILE)

TSZSRF(IEF) = total bits for representing secondary relationships in each occurrence of elementary file IEF.

$$= \sum_{ILE=1}^{GFNLE} INDFLE(IEF,ILE)*SECFNC(ILE)$$

where: SECFNC(ILE)=INDSEC(ILE)* TLEFNC(ILE)

TSZDEF(IEF) = total bits for representing definition in each occurrence of elementary file IEF.

$$= \sum_{ILE=1}^{GFNLE} INDFLE(IEF,ILE)*DEFFNC(ILE)$$

where: DEFFNC(ILE)=INDEFN(ILE)* TLEFNC(ILE)

TSZIDF(IEF) = total bits for representing data instances in each occurrence of elementary file IEF.

$$= \sum_{ILE=1}^{GFNLE} INDFLE(IEF,ILE)*DATFNC(ILE)$$

where: DATFNC(ILE)=INDATA(ILE)*TLEBIT(ILE)

These values are summed to yield totals for non-overhead portions of all LE's in each elementary file.

SNFLS(IEF) = total size in bits for all logical entries representing all functions (primary relationships, secondary relationships, definition, and data instances) in elementary file IEF.

= TSZPRF(IEF) + TSZSRF(IEF) + TSZDEF(IEF) + TSZIDF(IEF)

= TOTFNC

A similar procedure allows counting of logical entries by elementary file.

SNLEF(IEF) = total number of occurrences for all logical entry types in elementary file IEF.

$$= \sum_{ILE=1}^{GFNLE} INDFLE(IEF,ILE)*SNLEO(ILE)$$

FORTRAN functions are invoked to determine elementary file overhead, number of EF occurrences, and EF fixed size. These functions provide mechanisms for invoking intrinsic and optional parameter functions; see the "Parameter Functions" section at the end of this chapter.

IEFFOH(IEF) = overhead in bits for each occurrence of elementary file IEF; does not include overhead for specific LE types or for EF logical entries, and does not consider specified fixed size for elementary file IEF; function defined over parameter EFFOH(IEF).

SNEFO(ILE) = number of occurrences for elementary file IEF.

= EFOCCR(IEF); function defined over parameters EFNOC(IEF), EFRFO(IEF), and EFRFQ(IEF).

IEFIXZ(IEF) = fixed size in bits for each occurrence of elementary file IEF; includes all logical entries and all overhead; function defined, over parameter EFIXZ(IEF).

Using the variables defined above, total EF overhead and elementary file sizes are calculated.

SNFOH(IEF) = overhead in bits for all occurrences of elementary file IEF; includes all LE and EF overhead.

= (IEFIXZ(IEF)-SNFLS(IEF))<==>EFIXZ(IEF)=Ø

Otherwise:

= IEFFOH(IEF) * SNEFO(IEF) + LEFLEO(IEF)

Where:

LEFLEO(IEF) = total overhead for LE's in EF

$$= \sum_{ILE=1}^{GFNLE} INDFLE(IEF,ILE)*TLEOHD(ILE)$$

TEFSIZ(IEF) = total size in bits for all occurrences of elementary file IEF.

= IEFIXZ(IEF)*SNEFO(IEF) <==> EFIXZ(IEF)=Ø

Otherwise:

= SNFLS(IEF) + SNFOH(IEF)

EFSIZE(IEF) = size in bits for each occurrence of elementary file IEF.

= TEFSIZ(IEF)/SNEFO(IEF)

IBIOUS(IEF) = total size in basic I/O units for all occurrences of elementary file IEF.

= INTUP(EFSIZE(IEF)/BIUBIT)*SNEFO(IEF)

Finally, totals for the entire database are obtained by summing the results derived above for specific elementary files across all EF's in the database.

INLETD = total number of logical entry types in all elementary files

$$= \sum_{IEF=1}^{GFNEF} INLETF(IEF) = GFNLE$$

PSTLE = total number of occurrences for all logical entries in all elementary files.

$$= \sum_{IEF=1}^{GFNEF} SNLEF(IEF)$$

PSEFZ = total size in bits of all defined elementary files; does not include multiple elementary files.

$$= \sum_{IEF=1}^{GFNEF} EFSIZE(IEF)$$

PSTEF = total number of elementary files in DB including all elementary file occurrences.

$$= \sum_{IEF=1}^{GFNEF} SNEFO(IEF)$$

PSTSZ = total size in bits for all elementary files including all elementary file occurrences in DB.

$$= \sum_{IEF=1}^{GFNEF} TEFSIZ(IEF)$$

PSTPR = total bits for representing primary relationships in all occurrences of all elementary files in DB.

$$= \sum_{IEF=1}^{GFNEF} TSZPRF(IEF)$$

PSTSR = total bits for representing secondary relationships in all occurrences of all elementary files in DB.

$$= \sum_{IEF=1}^{GFNEF} TSZSRF(IEF)$$

PSTDE = total bits for representing definition in all occurrences of all elementary files in DB.

$$= \sum_{IEF=1}^{GFNEF} TSZDEF(IEF)$$

PSTID = total bits for representing data instances in all occurrences of all elementary files in DB.

$$= \sum_{IEF=1}^{GFNEF} TSZIDF(IEF)$$

PSTFO = total overhead in bits for all occurrences of all elementary files in DB.

$$= \sum_{IEF=1}^{GFNEF} TSZOHF(IEF)$$

TOTLES = total bits for all occurrences of all logical entry types in DB; includes logical entry and elementary file logical entry overhead, but does not include file overhead.

$$= \sum_{ILE=1}^{GFNLE} TLEBIT(ILE)$$

ITBIOU = total size in basic I/O units for all occurrences of all elementary files in DB.

$$= \sum_{IEF=1}^{GFNEF} IBIOUS(IEF)$$

7.4 Response Estimation

Unlike database size estimates that are invariant for a given parameter set, response time predictions reflect sequences of pseudo queries input by the on-line SPPM user. The response modeler must determine the event sequence necessary for responding to specific user queries. The complexities are procedural rather than mathematical; that is, once the sequence of events is determined, it is a relatively simple bookkeeping problem to accumulate processing and I/O times for each event. Consequently, the following paragraphs describing mathematical relationships embedded in response time estimation programs do not consider query analysis and event selection problems that are solved procedurally. These important and complex functions can be determined only by a careful review of the SPPM source code.

The response time estimation modeler produces two on-line summaries following each query, and two detailed analysis reports covering an entire query sequence input during a response estimation session. Hardcopy output includes reproduction of PSP pseudo query inputs and corresponding on-line summaries as well as detailed session analysis reports. One detailed report and one summary display estimate response time in milliseconds. The other two outputs are concerned with I/O activity and time requirements.

Response time estimates are derived by executing sequences of surrogate programs corresponding to PSP routines that would be invoked to answer an input query. Each surrogate program in the response modeler posts time and input/output accumulators. It is these accumulated values that appear on the summary and detailed analysis outputs.

The following paragraphs present mathematical relationships embedded in response time estimation programs. First, mechanisms for posting response estimates are described. Then, estimates for I/O and total query response times are derived in terms of model parameters and output variables.

7.4.1 Posting response estimates. Response modeler surrogates for PSP procedures update I/O and response time accumulators. Separate update mechanisms correspond to each of the three following types of accumulators:

* processing estimates for set processor procedures,

* I/O estimates for elementary files, and

* I/O time estimates for set processor procedures.

Throughout the response modeler, variables representing processor power and procedure repetition are used for calculating time requirements.

POWER = coefficient of processor power

= 1/ENPPI

GRPII = pseudo process iteration indicator; initialized to 1.

where:

GRPII > 1 ==> procedures invoked are replicated GRPII times.

Using these variables, each of the three posting mechanisms is described.

Processing estimates for set processor procedures The K-th posting of processor time and execution counts for N executions of set processor function IFN is summarized below.

NTIMES(IFN,N,K) = total number of pseudo executions for K-th posting for set processor function IFN.

= N * GRPII

PRNEX(IFN) = total number of pseudo executions for set processor function IFN (all postings).

$$= \sum_{\text{all } K} \text{NTIMES(IFN,K)}$$

TIMEST(IFN,K) = total estimated processor time in milliseconds for K-th posting for set processor function IFN.

= NTIMES(IFN,K) * POWER * (MSTIME + FNMOD(IFN)*MSTIME)

Where: MSTIME = FNPRC(IFN) parameter value, or is specified by posting procedure.

PRTPR(IFN) = total processor time in milliseconds for set processor function IFN (all postings).

$$= \sum_{\text{all } K} \text{TIMEST(IFN,K)}$$

Processor time estimates for all functions are maintained in accumulators GRTPQ and GRTPR. Given the following artificial variable,

QKIND(Q,K) = indicator of relationship between K-th posting and Q-th query

= 1 <==> K-th posting is the Q-th query
Ø otherwise

we define total accumulations as follows.

GRTPQ(Q) = total processor time in milliseconds for Q-th query in response estimation session.

$$= \sum_{\text{all } K} QKIND(Q,K) * TIMEST(IFN,K)$$

for all IFN invoked for query Q.

GRTPR = total processor time in milliseconds for entire response estimation session.

$$= \sum_{\text{all } Q} GRTPQ(Q)$$

<u>I/O estimates for elementary files</u> I/O accumulators for elementary files are posted by I/O estimation routines. For the J-th posting of I/O estimation results, the following variables are defined.

IEFNAM(J) = elementary file name for the J-th I/O posting.

IOP(J) = I/O operation for the J-th I/O posting.

= 'RD', 'WT', 'OP', 'CL', or 'DE'

NOP(J) = number of physical I/O requests for J-th I/O posting.

NOBIOU(J) = number of basic I/O units transferred for J-th I/O posting.

IOACC(J) = I/O access time for J-th I/O posting.

IOTRN(J) = I/O transfer time for J-th I/O posting.

IOOTH(J) = other I/O time for J-th I/O posting.

Then total access, transfer and other times and total basic I/O units transferred are posted in the following manner.

ITOTIM(J,IEF) = total I/O time for J-th I/O posting for elementary file IEF reference

= (IOACC(J) + IOTRN(J) + IOOTH(J)) * POWER * GRPPI

Where: IEF = IEFSUB(IEFNAM(J))

IETIM(IEF) = total I/O time for elementary file IEF (all postings).

$$= \sum_{\text{all } J} \text{ITOTIM(J,IEF)}$$

GRACQ(Q) = total I/O access time for Q-th query in response estimation session.

$$= \sum_{\text{all } J} \text{IOACC(J)} * \text{POWER} * \text{GRPPI}$$

GRTNQ(Q) = total I/O transfer time for Q-th query in response estimation session.

$$= \sum_{\text{all } J} \text{IOTRAN(J)} * \text{POWER} * \text{GRPPI}$$

GROTQ(Q) = total other I/O time for Q-th query in response estimation session.

$$= \sum_{\text{all } J} \text{IOOTH(J)} * \text{POWER} * \text{GRPPI}$$

GRBTQ(Q) = total basic I/O units transferred for Q-th query in response estimation session.

$$= \sum_{\text{all } J} \text{NOBIOU(J)*GRPII}$$

Numbers of physical I/O requests are accumulated for I/O operations and for elementary files. First, the following indicator variables are defined to simplify mathematical representations.

IOPRD(J) = read operation indicator for J-th I/O posting.

= 1 <==> IOP(J) = "RD"
Ø otherwise

IOPWT(J) = write operations indicator for J-th I/O posting.

= 1 <==> IOP(J) = "WT"
Ø otherwise

IOPOP(J) = open operation indicator for J-th I/O posting.

= 1 <==> IOP(J) ="OP"
Ø otherwise

IOPCL(J) = close operation indicator for J-th I/O posting.

= 1 <==> IOP(J) = "CL"
Ø otherwise

IOPDE(J) = delete operation indicator for J-th I/O posting.

= 1 <==> IOP(J) = "DE"
Ø otherwise

IEFIO(IEF,J) = indicator for J-th posting of elementary file IEF I/O.

= 1 <==> IEF = IEFSUB(IEFNAM(J))

Then, numbers of physical I/O requests are accumulated in the following manner.

GRRRQ(Q) = total number of physical read requests to Q-th query in response estimation session.

$$= \sum_{\text{all } J} IOPRD(J)*NOP(J)$$

IENRR(IEF) = total number of physical read requests for elementary file IEF for entire response estimation session.

$$= \sum_{\text{all } J} \text{IEFIO(IEF,J)*IOPRD(J)}$$

GRWRQ(Q) = total number of physical write requests for Q-th query in response estimation session.

$$= \sum_{\text{all } J} \text{IOPWT(J)*NOP(J)}$$

IENWR(IEF) = total number of physical write requests for elementary file IEF for entire response estimation session.

$$= \sum_{\text{all } J} \text{IEFIO(IEF,J)*IOPWT(J)}$$

GROPQ(Q) = total number of open requests for Q-th query in response estimation session.

$$= \sum_{\text{all } J} \text{IOPOP(J)*NOP(J)}$$

GPCLQ(Q) = total number of close requests for Q-th query in response estimation session.

$$= \sum_{\text{all } J} \text{IOPCL(J)*NOP(J)}$$

GRDEQ(Q) = total number of delete requests for Q-th query in response estimation session.

$$= \sum_{\text{all } J} \text{IOPDE(J)*NOP(J)}$$

IEØTH(IEF) = total number of I/O requests other than RD or WT for elementary file IEF for entire response estimation session.

$$= \sum_{\text{all } J} IEFIO(IEF,J)*IOPOTH$$

Where:

IOPOTH = IOPOP(J) + IOPCL(J) + IOPDE(J)

<u>I/O estimates for set processor functions</u> I/O time estimates are accumulated for set processor functions requesting I/O operations. For the I-th posting of I/O time to set processor functions the following variables are defined.

IFNAME(I) = name of set processor function for I-th posting of I/O time.

IOTIME(I) = I/O time in milliseconds for I-th posting of I/O time to set processor functions .

IFNIO(IFN,I) = indicator of relationship between I-th posting of I/O time and set processor function IFN.

= 1 <==> IFN = IFNSUB(IFNAME(I))
Ø otherwise

Then, I/O time accumulation for set processor functions is as follows.

PRTIO(IFN) = total I/O time in milliseconds for set processor function IFN

$$= \sum_{\text{all } I} IFNIO(IFN,I)*(IOTIME(I)*GRPII)$$

<u>7.4.2 I/O estimation.</u> Figure 7.4 contains copies of the I/O on-line summary and detailed session analysis reports with variable names inserted in brackets under data entries. Accumulation variables have all been defined in the preceding section. Extensions and totals are defined below.

I/O SUMMARY

DESC	QUERY		SESSION	
NO PHYSICAL READS	2 [GRRRQ]		7 [GRTRR]	
NO PHYSICAL WRITES	Ø [GRWRQ]		Ø [GRTWR]	
NO OTHER I/O'S				
OPEN	Ø [GROPQ]		Ø [GRTOP]	
CLOSE	Ø [GRCLQ]		Ø [GRTCL]	
DELETE	Ø [GRDEQ]		Ø [GRTDE]	
NO BIOU'S TRANS	1Ø [GRBTQ]		26 [GRBTR]	
ACCESS TIME	46.ØØ [GRACQ]	/ 8.27%	161.ØØ [GRACC]	/ 1Ø.47%
TRANSFER TIME	51Ø.ØØ [GRTNQ]	/ 91.73%	1326.ØØ [GRTRN]	/ 86.27%
OTHER I/O TIME	Ø.ØØ [GROTQ]	/ Ø.ØØ%	5Ø.ØØ [GROTI]	/ 3.25%
TOTAL TIME	556.ØØ [GRTIQ]		1537.ØØ [GRTIO]	

SESSION I/O ANALYSIS BY ELEMENTARY FILE

E-FILE	NO. PHYSICAL I/O REQUESTS			NO BIOU'S	TOTAL I/O TIME	
	READ [IENRR]	WRITE [IENWR]	OTHER [IEOTH]	TRAN [IEBTR]	MS [IETIM]	%
ELMNT	6	Ø	Ø	3Ø	1668.ØØ	79.69
SRAVI	1	Ø	Ø	2	15Ø.ØØ	7.17
SINVS	1	Ø	Ø	2	15Ø.ØØ	7.17
SWORK	1	Ø	Ø	2	125.ØØ	5.97
TOTAL	9 [GRTRR]	Ø [GRTWR]	Ø [GRTDE]	36 [GRBTR]	2Ø93.ØØ [GRTIO]	1ØØ.ØØ

Figure 7.4

I/O summary

GRTIQ(Q) = total I/O time in milliseconds for Q-th query in response estimation session.

= GRACQ(Q) + GRTNQ(Q) + GROTQ(Q)

GRACQP(Q) = access time as a percentage of total I/O time for Q-th query in response estimation session.

= GRACQ(Q)/GRTIQ(Q)

For each query level accumulator there is a corresponding session total; session totals and percentages appearing on the I/O summary are defined below.

GRTRR = total number of read requests for session.

$$= \sum_{\text{all } Q} GRRRQ(Q)$$

GRTWR = total number of write requests for session.

$$= \sum_{\text{all } Q} GRWRQ(Q)$$

GRTOP = total number of file opens for session.

$$= \sum_{\text{all } Q} GROPQ(Q)$$

GRTCL = total number of file closes for session.

$$= \sum_{\text{all } Q} GRCLQ(Q)$$

GRTDE = total number of file deletes for session.

$$= \sum_{\text{all } Q} GRDEQ(Q)$$

GRBTR = total number of BIOU's transferred for session.

$$= \sum_{\text{all } Q} GRBTQ(Q)$$

GRACC = total I/O access time for session.

$$= \sum_{\text{all } Q} GRACQ(Q)$$

GRTRN = total I/O transfer time for session.

$$= \sum_{\text{all } Q} GRTNQ(Q)$$

GROTI = total other I/O time for session.

$$= \sum_{\text{all } Q} GRTIQ(Q)$$

GRACCP = access time as a percentage of total I/O time for session.

= GRACC/GRTIO

GRTRNP = transfer time as a percentage of total I/O time for session.

= GRTRN/GRTIO

GROTIP = other I/O time as a percentage of total I/O time for session.

= GROTI/GRTIO

Session I/O analysis by elementary file

EFNAM(IEF) = name of elementary file IEF (by definition).

IETIM(IEF) = total I/O time in milliseconds for elementary file IEF for entire response estimation session.

= IENRR(IEF) + IENWR(IEF) + IEØTH(IEF)

IETIMP(IEF) = total I/O time for elementary file IEF as a percentage of total session I/O time

= IETIM(IEF)/GRTIO

IIEOTH = total number of I/O requests other than RD or WT for entire response estimation session

= GRTOP + GRTCL + GRTDE

$$= \sum_{IEF=1}^{GFNEF} IEOTH(IEF)$$

Determining I/O estimates

The Positional Set Processor database management system that is the object of this SPPM modeling effort performs all direct access I/O through a single, generalized input-output procedure. I/O requests are in terms of Smallest Addressable Units (SAU's/e.g. words). Transfer of data to and from secondary storage is carried out in Basic I/O units (BIOU's/e.g. sectors). The response modeler mimics this PSP high level I/O interface.

I/O time estimates for open, close and delete operations are simply taken from parameters ENIOP, ENICL and ENIDE respectively. The model, like the PSP prototype, automatically "opens" a file that is to be read or written if it is not already open.

Read and write time estimates have three components: access time transfer time, and software time. Model parameters specify read and write access and transfer times in milliseconds. The problem is to determine coefficients for these model variables given the following arguments describing specific I/O requirements.

IOP = I/O operation; RD, WT, OP, CL or DE.

IOEF = elementary file for I/O operation.

IOTRQ = the amount of data in SAU's requested for each I/O transfer; one or more physical I/O's may be required to satisfy this requirement.

ISTART = the starting SAU in elementary file IOEF for first I/O transfer for this requirement.

TOTDAT = the total amount of data in SAU's for all I/O transfers for this requirement.

Note that a single model request of size TOTDAT may represent multiple sequential transfers of IOTRQ SAU's starting in SAU ISTART.

The I/O estimator, like the PSP routine and I/O software it mimics, handles various complexities.

* Write requests that do not fall on BIOU beginning and ending boundaries require preliminary reads.

* Read and write requests that do not fall on BIOU boundaries use an intermediate buffer or require multiple I/O transfers.

* I/O requests for elementary files that have not been previously referenced are opened before they are read or written.

* Look ahead buffering is employed; physical read requests fill the buffer I/O software buffers.

Calculations to determine numbers of accesses and BIOU transfers for read and write requests are now summarized. First, the following variables are defined.

IDIVSR = greatest common divisor of I/O software buffer for elementary file IEF and maximum number of BIOU's that can be transferred with single access.

= IGCD(IEBUF(IEF),ENIOM)

Then, using the term "cluster" to refer to ENIOM BIOU's and assuming that ISTART is both a buffer and a cluster starting SAU, the number of accesses required is computed.

NBBHCB = number of buffers between hits on cluster boundaries.

= ENIOM/IDIVSR

NBTRW = number of buffers to be read/written.

= INTUP(TOTDAT/)IEFBUF(IEF))*ENBIU))

NCTRW = number of clusters to be read/written.

= INTUP(TOTDAT/(ENIOM*ENBIU))

NCFOBB = number of clusters falling on buffer boundaries.

= INTUP(NBTRW/NBBHCB)

TNPA = total number of positioning actions (arm movement and latency).

= NBTRW + NCTRW - NCFOBB

Finally, the number of BIOU transfers required for read operations is determined.

NBIOUT = number of basic I/O unit transfers for I/O requirements

= INTUP(TOTDAT/ENBIU)

Additional complexities for write requests requiring more than one logical access are not reflected in the above equations. An iterative, augmented procedure is used when I/O requests do not start on both cluster and buffer boundaries.

7.4.3 Response time estimation outputs. Response summary and detailed analysis reports appear in Figure 7.5; again, most of the variable names inserted in brackets under data entries have been previously defined. Extensions and total definitions are as follows.

Response summary

GRTOQ(Q) = total overhead in milliseconds of response time for Q-th query in response estimation session.

= GRTIQ(Q) + GRTPQ(Q) * (1-ENLOD)

RESPONSE SUMMARY REPORT

RESPONSE SUMMARY

DESC	QUERY MS	QUERY %	SESSION MS	SESSION %
I/O	556.ØØ [GRTIQ]	6.15	1537.Ø [GRTIO]	1Ø.47
PROCESSING	8478.Ø [GRTPQ]	93.85	13137. [GRTPR]	89.53
OVERHEAD	Ø.ØØØØØE+ØØ [GRREQ]	Ø.ØØ	Ø.ØØØØØE+ØØ [GRTOH]	Ø.ØØ
RESPONSE	9Ø34.Ø [GRREQ]	1ØØ.ØØ	14674. [GRRES]	1ØØ.ØØ

Figure 7.5a

RESPONSE DETAILED ANALYSIS REPORT

SESSION RESPONSE ANALYSIS BY DATABASE FUNCTION

DATABASE FUNCTION [FNNAM]	NO EXEC [PRNEX]	TIME IN MS PROCESS [PRTPR]	TIME IN MS I/O [PRTIO]	TOTAL TIME MS [PRTIP]	TOTAL TIME %
SSAVE	23	184.ØØ	8179.ØØ	8363.ØØ	31.13
SCOPY	2Ø	556Ø.ØØ	Ø.ØØ	556Ø.ØØ	15.Ø2
UNION	1Ø	4274.ØØ	Ø.ØØ	4274.ØØ	11.46
RLCMP	5	2745.ØØ	Ø.ØØ	2745.ØØ	6.86
S9PRO	3	25Ø2.ØØ	Ø.ØØ	25Ø2.ØØ	6.81
RGSTR	11	451.ØØ	174Ø.ØØ	2191.ØØ	6.47
A7SRC	11	1474.ØØ	Ø.ØØ	1474.ØØ	4.37
ALLOC	44	352.ØØ	76Ø.ØØ	1112.ØØ	4.2Ø
INTRS	2	1Ø82.ØØ	Ø.ØØ	1Ø82.ØØ	4.Ø7
M9COM	4	944.ØØ	Ø.ØØ	944.ØØ	4.Ø1
SADD1	4	81Ø.ØØ	Ø.ØØ	81Ø.ØØ	3.5Ø
S4IDX	8	64Ø.ØØ	Ø.ØØ	64Ø.ØØ	2.76
S4MOV	16	2Ø8.ØØ	162.ØØ	37Ø.ØØ	1.6Ø
S4SUB	3	159.ØØ	Ø.ØØ	159.ØØ	Ø.69
S4EVA	3	9Ø.ØØ	Ø.ØØ	9Ø.ØØ	Ø.39
A7TRA	11	88.ØØ	Ø.ØØ	88.ØØ	Ø.38
S9PTR	3	84.ØØ	Ø.ØØ	84.ØØ	Ø.36
SDEST	15	75.ØØ	Ø.ØØ	75.ØØ	Ø.32
DALOC	1Ø	5Ø.ØØ	Ø.ØØ	5Ø.ØØ	Ø.22
S4CHK	3	36.ØØ	Ø.ØØ	36.ØØ	Ø.16
M9PID	3	36.ØØ	Ø.ØØ	36.ØØ	Ø.16
M9GTS	3	3Ø.ØØ	Ø.ØØ	3Ø.ØØ	Ø.13
M9ALO	3	27.ØØ	Ø.ØØ	27.ØØ	Ø.12
S4OPR	5	25.ØØ	Ø.ØØ	25.ØØ	Ø.11
S4BLD	5	25.ØØ	Ø.ØØ	25.ØØ	Ø.11
S4SRC	8	24.ØØ	Ø.ØØ	24.ØØ	Ø.1Ø
M9GTL	3	12.ØØ	Ø.ØØ	12.ØØ	Ø.Ø5
M9ISO	4	12.ØØ	Ø.ØØ	12.ØØ	Ø.Ø5
(OTHER)	Ø [PRONX]	Ø.ØØ [PROPR]	Ø.ØØ [PROIO]	Ø.ØØ [PROIP]	Ø.ØØ
TOTAL	243	21999.ØØ [GRTPR]	1Ø841.ØØ [GRTIO]	3284Ø.ØØ [GRTIP]	1ØØ.ØØ
OVERHEAD				Ø.ØØ [GRTOH]	Ø.ØØ
TOTAL RESPONSE				3284Ø.ØØ [GRRES]	1ØØ.ØØ

Figure 7.5b

GRREQ(Q) = total response time in milliseconds for Q-th query in response estimation session.

= GRTIQ(Q) + GRTPQ(Q) + GRTOQ(Q)

GRTIQP(Q) = total I/O time as a percentage of total response time for Q-th query in response estimation session.

= GRTIQ(Q) / GRREQ(Q)

GRTPQP(Q) = total processing time as a percentage of total response time for Q-th query in response estimation session.

= GRTPQ(Q) / GRREQ(Q)

GRTOQP(Q) = total overhead as a percentage of total response time for Q-th query in response estimation session.

= GRTOQ(Q) / GRREQ(Q)

PRTEX = total number of executions for all database functions for response estimation session.

$$= \sum_{IFN=1}^{GSNFN} PRNEX(IFN)$$

Finally, session accumulators and percentages corresponding to query level variables are defined.

GRTPR = total quiescent system procession time for session.

$$= \sum_{all\ Q} GRTPQ(Q)$$

GRTIP = total quiescent system response time in milliseconds for all database functions invoked during entire response estimation session.

= GRTPR + GRTIO

PRTIPP(IFN) = total time for database function IFN as a percentage of total quiescent system response time for entire response estimation session.

= PRTIP(IFN) / GRTIP

GRTOH = total system overhead for session.

$$= \sum_{\text{all } Q} GRTOQ(Q)$$

GRRES = total response time for session.

$$= \sum_{\text{all } Q} GRREQ(Q)$$

GRTIOP = I/O time as a percentage of total response time for session.

= GRTIO/GRRES

GRTPRP = processing time as a percentage of total response time for session.

= GRTPR/GRRES

GRTOHP = system overhead as a percentage of total response time for session.

= GRTOH/GRRES

<u>Session response analysis by database function</u>

FNNAM(IFN) = name of set processor function IFN(by definition).

PRTIP(IFN) = total quiescent system time in milliseconds for database function IFN.

= PRTPR(IFN) + PRTIO(IFN)

<u>7.4.4 Monte Carlo processes.</u> Monte carlo processes are used throughout the response modeler to determine specific quantities and references not directly derivable from parameter and query sequence inputs. While facilities have been provided for using other distributions, all Monte Carlo variables in the current SPPM implementation are selected from

the uniform distribution. Upper and lower bounds are determined from parameters, from size estimates, and from intermediate model variables. Monte Carlo estimates can be generated for both integer and floating point random variables. The model utilizes the DEC FORTRAN-1Ø random number generator function, RAN, for Monte Carlo calculations.

<u>7.4.5 Determination of set cardinalities.</u> Determination of set cardinalities is an essential part of the response estimation process. Set cardinalities are determined from parameter inputs; from on-line user responses, and from Monte Carlo estimation procedures. In order to discuss the determination of set cardinalities we define the following:

ISX = set representation X; X=1,2,3 ... M.

CARD(ISX) = cardinality for set ISX; that is, the number of elements in set ISX.

<u>Cardinalities for sets satisfying elementary conditions</u> A query containing a complex predicate (e.g., in the PSP SUBX command) can be viewed as one or more elementary conditions connected by boolean operators. Each elementary condition takes the form:

<SUBJECT> <RELATIVE OPERATOR> <OBJECT>

where: <SUBJECT> = attribute name

<RELATIVE OPERATOR> = EQ,NE,GT,GE,LT,or LE.

<OBJECT> = value or attribute name

The current version of the SPPM limits elementary conditions to those susceptible to solution using secondary indices. Currently, secondary indices are not used for answering PSP queries when elementary condition <OBJECT> entries are attribute names. Thus, only simple conditions with <OBJECT> = VALUE have been considered in the preliminary SPPM implementation. Cardinalities for sets satisfying elementary conditions are specified by on-line users or optionally are determined by the model. In either case, upper and lower bounds are determined as follows: Given elementary condition,

RELNAM.ATTNAM OPERATOR <VALUE>

we determine boundaries for the cardinality of the solution set in the following manner.

IRL = IRLSUB(RELNAM)

IAT = IATSUB(ATTNAM)

UNIVAL(IRL,IAT) = number of unique instances of attribute IAT in relation IRL.

= RLCRD(IRL) <==> DMNVL(IDM) > RLCRD(IRL)
DMNVL(IDM) otherwise

where: IDM = IDMSUB(ATDOM(IAT))

MINCRD(C) = minimum cardinality for solution set for elementary condition C.

= Ø <==> OPR ≠ EQ and NE
(UNIVAL(IRL,IAT) - 1) otherwise

MAXCRD(C) = maximum cardinality for solution set for elementary condition C.

= RLCRD(IRL) <==> OPR ≠ EQ and NE
(RLCRD(IRL)-(Unival(IRL,IAT)-1) otherwise

On-line users may provide set cardinalities within these boundaries. Model determination of cardinalities is through Monte Carlo techniques using a uniform distribution within the boundaries defined above for conditions with operators LT, LE, GT or GE.

CRDEST(C) = estimated cardinality of solution set for elementary condition C.

$$MINCRD(C) \le CRDEST(C)=f(U) \le MAXCRD(C)$$

For operators EQ and NE, the average instance frequency is used to estimate solution set cardinalities.

AVGPTR(IRL,IAT) = average number of instances for attribute IAT in relation IRL; also, the cardinality of the average secondary index pointer for attribute IAT in relation IRL.

= RLCRD(IRL)/UNIVAL(IRL,IAT)

CRDEST(C) = RLCRD(IRL)-AVGPTR(IRL,IAT) <==> OPR = NE

AVGPTR(IRL,IAT) <==> OPR = EQ

Cardinalities for sets resulting from boolean operations

Cardinalities for sets resulting from boolean operations are determined by Monte Carlo processes using the uniform distribution. Minimum (MINCRD) and maximum (MAXCRD) cardinality boundaries for sets resulting from each of the four boolean operators are defined in the following manner.

* UNION(IS1,IS2,IS3) ==> IS3 = IS1 U IS2

  MINCRD(IS3) = MAXINT(CARD(IS1),CARD(IS2))
  MAXCRD(IS3) = CARD(IS1) + CARD(IS2)

* INTRS(IS1,IS2,IS3) ==> IS3 = IS1 $\cap$ IS2

  MINCRD(IS3) = Ø
  MAXCRD(IS3) = MININT(CARD(IS1),CARD(IS2))

* XUNSD(IS1,IS2,IS3) ==> IS3 = (IS1 U IS2) - (IS1 $\cap$ IS2)

  MINCRD(IS3) = |CARD(IS1) - CARD(IS2)|
  MAXCRD(IS3) = CARD(IS1) + CARD(IS2)

* RLCMP(IS1,IS2,IS3) ==> IS3 = IS1 - IS2

  MINCRD(IS3) = O <==> CARD(IS2) $\geq$ CARD(IS1)
  CARD(IS1) - CARD(IS2) otherwise
  MAXCRD(IS3) = CARD(IS1)

7.5 Bit-string Size Estimation

The Positional Set Processor prototype represents sets as compacted bit strings. The Quatree compaction algorithm described by Hardgrave [HARD73a, HARD76a] represents sets as a multi-leveled tree of n-bit packets. Both size and response modelers must estimate sizes for bit-string representations of sets.

An estimate of the size of bit-string set representation ISR is determined using model parameters BSPKS and BSQLV, for the packet size and the number of Quatree levels respectively, and the following arguments.

SRCRD(ISR) = cardinality of set representation ISE; that is, the number of elements in the set.

SRNGE(ISR) = range over which set representation ISR is defined; that is the ordinal of the largest possible "on" bit in the logical bit string set representation.

Then the number of tree levels required for the specified range is determined.

NLEVL = INTUP(LOG(SRNGE(ISR))/LOG(BSPKS))

For each level LL the number of packets for the given range is determined.

SUBPK(LL) = total number of packets at level LL for range where levels are numbered starting with one at the root.

= INTUP(SRNGE(ISR)/(BSPKS**LL))

Then, assuming that set elements are evenly distributed throughout the range, the following are calculated.

MAXPK = maximum number of packets for subtree required to represent set for given cardinality, range, and packet size.

$$= \sum_{LL=1}^{NLEVL} \text{MININT(SRCRD(ISR),SUBPK(LL))}$$

BSBITS = number of bits required to represent set of given cardinality and range for specified packet size and number of levels.

= (MAXPK + BSQLV - NLEVL)*BSPKS

7.6 Parameter Functions

The SPPM utilizes two types of parameter functions: intrinsic functions that define elementary file and logical entry occurrence frequencies, and optional functions that can be specified for other parameters describing secondary storage utilization. These two function classes are discussed in the following paragraphs.

7.6.1 Intrinsic occurrence frequency functions. Occurrence frequencies for elementary files and logical entities are determined from parameter tuples. For instance, the number of occurrences for elementary file IEF would be:

<EFNOC(IEF)> times for each <EFREO(IEF)> in <EFREQ(IEF)>

Where: EFNOC(IEF) = integer number of occurrences
EFREO(IEF) = character string selected from list appearing in Figure 7.6.
EFREQ(IEF) = reference to relational entities defined in parameter set.

e.g., <2> times for each <AT>tribute in relation <PERSN>

7.6.2 Optional parameter functions. The SPPPM allows the user to invoke complex functions in the form of FORTRAN procedures when simple parameter constants are not sufficient for describing an object database management system. Optional parameter functions are represented in the parameter set by an asterisk (*) followed by (up to) four characters. FORTRAN function subroutines that recognize and invoke defined parameter functions must be modified when new functions are defined (FORTRAN requires that all program references be defined at load time). Figure 7.7 lists all parameters that can be functionally specified, their corresponding FORTRAN function subroutines, and all functions defined for the preliminary SPPM implementation.

RELATIONAL ENTITY OCCURRENCE INDICATORS

REO	DESCRIPTION
<null>	occurence specified in parameter XXNOC(IXX) where : XX = EF or LE
RL	relation
RLX	indexed relation
TU	tuple
DM	domain
AT	attribute
ATX	indexed attribute
AI	attribute instance
AID	attribute instance unique within DB
AIR	attribute instance unique within relation
AIA	attribute instance unique within attribute
AIX	instance of an indexed attribute
AIXA	instance of an indexed attribute unique within attribute
LE	logical entry
EF	elementary file

Figure 7.6

OPTIONAL PARAMETER FUNCTIONS

PARAMETER	FORTRAN FUNCTION	DEFINED PARAMETER FUNCTIONS
EFIXZ(IEF)	IEFIXZ(EFNAM(IEF))	*LASH : fixed size for a hash table to be referenced with linear search and average number of probes = 3
EFFOH(IEF)	IEFFOH(EFNAM(IEF))	*AVOH : overhead for files written with 1Ø SAU headers and fixed length buffers of 128 SAU's for all occurences of each logical entry type
EFEOH(IEF)	IEFFOH(EFNAM(IEF), LENAM(ILE))	*FXLB : overhead for logical entry ILE in elementary file IEF written with fixed length logical entry buffers of 128 SAU's; i.e. each LE takes a multiple of 128 SAU's
LESIZ(ILE)	ILESIZ(LENAM(ILE))	*PBST : size for PSP bit string logical entry *IRBS : size for ISP bit string representing universal set for relation *IIBS : size for ISP bit string representing secondary index pointer set
LEOHD(ILE)	ILEOHD(LENAM(ILE))	(no functions currently defined)

Figure 7.7

8. MODEL EVALUATION

8.1 Introduction

The term operations research (OR) was coined during World War II to refer to an interdisciplinary, scientific approach for solving the very real problems of managing military operations. Today, numerous synonyms for operations research, including the currently favored management science, are commonly used to describe a scientific approach to problem solving for executive management [WAGN69 pp1-31]. While military problems no longer dominate the field, the practical problem solving orientation of operations research remains. Today, industrial and administrative as well as military decision making are addressed by operations research practitioners.

Because of the practical orientation that has been part of operations research since its inception, one would expect that the determination of the goodness of decision models would be an important and well developed part of OR procedures. But this aspect of modeling that we term (for lack of something better) model evaluation surprisingly is ignored by many textbooks and is given only cursory, philosophical treatment by others. This chapter considers several dimensions of the model evaluation problem, presents some specific thoughts on computer system models, and describes evaluation procedures for the set processor performance model.

8.2 Evaluation Phases

Model evaluation is a complex and multi-faceted process that is generally viewed as having several phases. Fishman and Kiviat [FISH67] identify three evaluation phases that are mentioned throughout the literature.

* Verification - insuring that the model behaves as the experimenter intends,

* Validation - testing the agreement between the behavior of the model and that of the real world system, and

* Problem Analysis - analyzing and interpreting data generated by experiments using the model.

Each of these phases is discussed briefly below.

8.2.1 Verification. All but the most trivial models today are realized in the form of algorithms (i.e., programs) for execution on high-speed digital computers. Verification is concerned with assuring the correctness of program realizations of models. This debugging, as it is called by computer scientists, is itself a complex and time consuming process. Program testing and correction procedures are still largely ad hoc; with the exception of extremely simple algorithms, we can not prove the correctness of a piece of computer source code [FIFE77, ELSP72, HANT76]. Recognizing the difficulty and magnitude of the verification task, it is not considered further here.

8.2.2 Validation. Model validation is concerned with determining how well important characteristics of a real world system are reflected in a model surrogate. Frequently the term validation is used to refer to the entire evaluation process. In the more limited sense in which it is used here, validation presupposes that verification of correctness for the program realization of the model has been accomplished.

Types of Validity Several aspects of the validation problem are addressed by the literature. Zeigler [ZEIG76] identifies three types of validity: replicative, predictive, and structural. A model is replicatively valid when its behavior matches data already acquired from the real system. Predictive validity is a stronger condition that exists when model data is derived before data from the real system confirms model predictions. The strongest form of validity defined by Zeigler is concerned with isomorphism between the model and the real system. Structural validity occurs when a model not only reproduces the real system behavior, but also reflects the manner in which the real system operates. Shannon [SHAN75] addresses the question of whether a model should be an isomorphic reflection of a real world system; he concludes that the question has been debated for years and is still unanswered today.

Validation Philosophies Shannon prefers to view validation as merely one aspect of scientific enquiry as suggested by Churchman [CHUR68]. In this context, he defines three extreme approaches to model development and validation.

* Rationalism - this modeling approach is based on the existence of premises of unquestionable truth that need not be explicitly proven. Acceptance of the premises and of the logic with which they are connected implies acceptance of the validity of the model. The most notable modern day examples of the rationalist approach are the urban and world models of Forrester [FORR69, FORR71].

* Empiricism - this modeling approach requires that all model components must be based on premises or assumptions that can be independently verified by experiment or analysis of empirical data.

* Absolute pragmatism - this modeling approach sees validation as being concerned with whether a model achieves the purpose for which it was developed. Thus, usefulness rather than truth determines validity.

Multi-stage utilitarian approach In practice, however, few modeling efforts reflect one of the philosophical extremes described above. Most often, a validation approach that combines aspects of all three philosophies is employed. Termed by Shannon the utilitarian approach, Naylor and Finger [NAYL67] describe a multi-stage validation process that represents such a philosophical compromise, falling in the middle of the pure philosophies as illustrated in Figure 8.1.

* STAGE 1 - determine that model building block components have face validity; that is, assure that basic components are reasonable and that assumptions make sense. This stage is a modified rationalist approach.

* STAGE 2 - empirical testing of model components and relationships; this is a modified empiricist approach, using statistical techniques such as tests of hypotheses, for testing assumptions, parameters and relationships.

* STAGE 3- matching model predictions to the behavior of the real world system; the ability of the model to predict is viewed as the most important indicator that the model satisfies the absolute pragmatist criteria of usefulness.

These three stages are applied iteratively throughout the model development and application process. Thus, a continuing spiral of modified rationalism and empiricism followed by absolute pragmatism occurs until an acceptable level of validation is achieved. The question of how much validation is enough is addressed in a subsequent section.

8.2.3 Problem analysis. The third part of the evaluation process is concerned with analyzing and correctly interpreting data produced by the model. Like verification, this evaluation phase is complex, and is itself the subject of study and entire treatises. A model that has been verified and validated can still result in bad decisions if model

VALIDATION PHILOSOPHIES

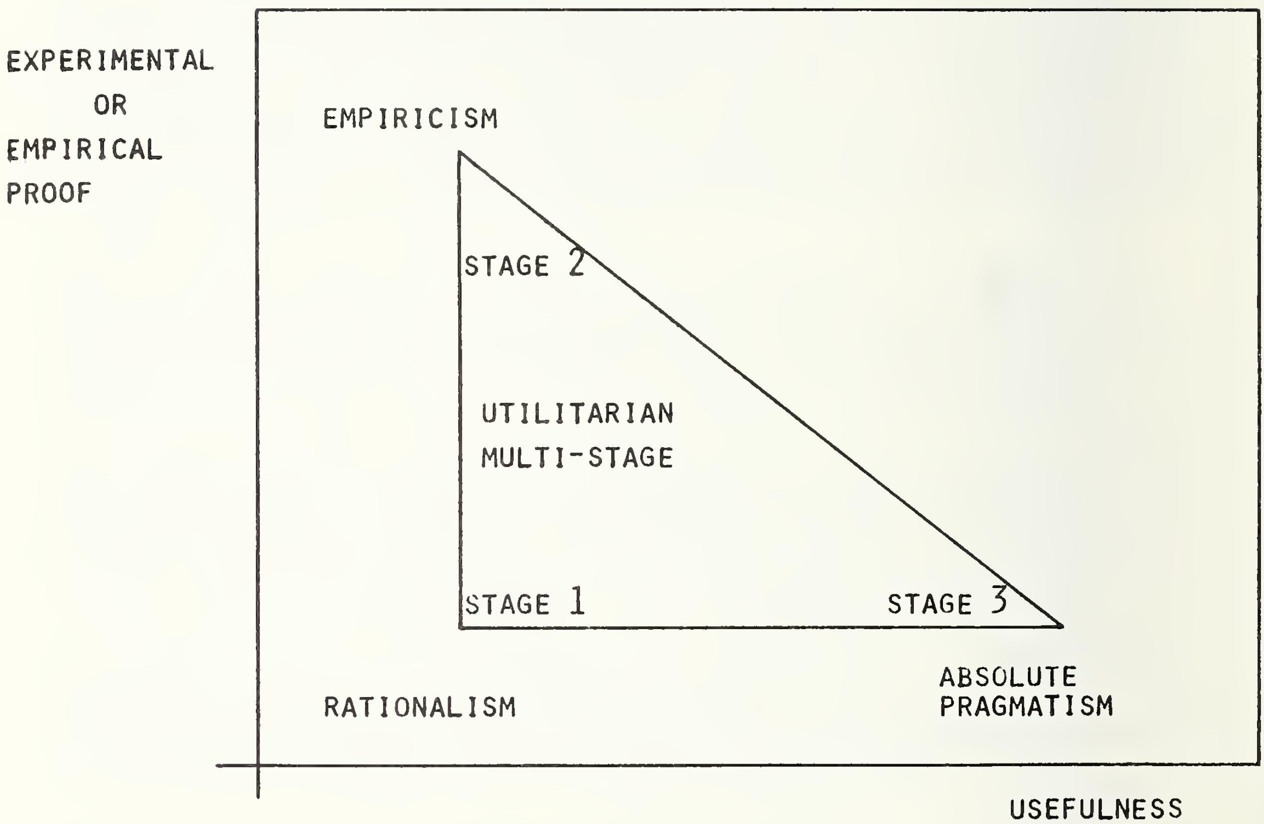


FIGURE 8.1

produced data are misunderstood or improperly used. Problem analysis must, therefore, be recognized as an important and integral part of the evaluation process.

8.3 Error Taxonomies

Errors uncovered by the model evaluation process can take several forms. Error taxonomies appear throughout the literature. Ferrari [FERR78 pp134-139] identifies three classes of inaccuracies:

* formulation errors - caused by a model that does not represent the real world system correctly or in sufficient detail,

* solution errors - caused by applying incorrect solution techniques to the model representation, and

* parameter errors - caused by the use of incorrect parameter values.

Shannon [SHAN75] lists five classes of errors that can lead to erroneous conclusions.

* Design errors

* Programming errors

* Data errors

* Procedural (model usage) errors, and

* Interpretation errors.

Within the framework of the three evaluation phases and the multi-stage utilitarian validation approach described above, Figure 8.2 is a synthesis of these error taxonomies and of Zeigler's validity classification. It is clear from this tabular analysis that these classifications differ in scope and emphasis; each views model evaluation from a slightly different perspective.

8.4 Acceptance Criteria

Model validity is often thought of as a binary characteristic, either the model is valid or it is not. This is an unfortunate misconception; proof of absolute validity may be neither theoretically nor economically feasible. The multi-stage utilitarian view of validation recognizes the concept of relative validity, with additional iterations

SYNTHESIS OF MODEL EVALUATION TERMINOLOGY

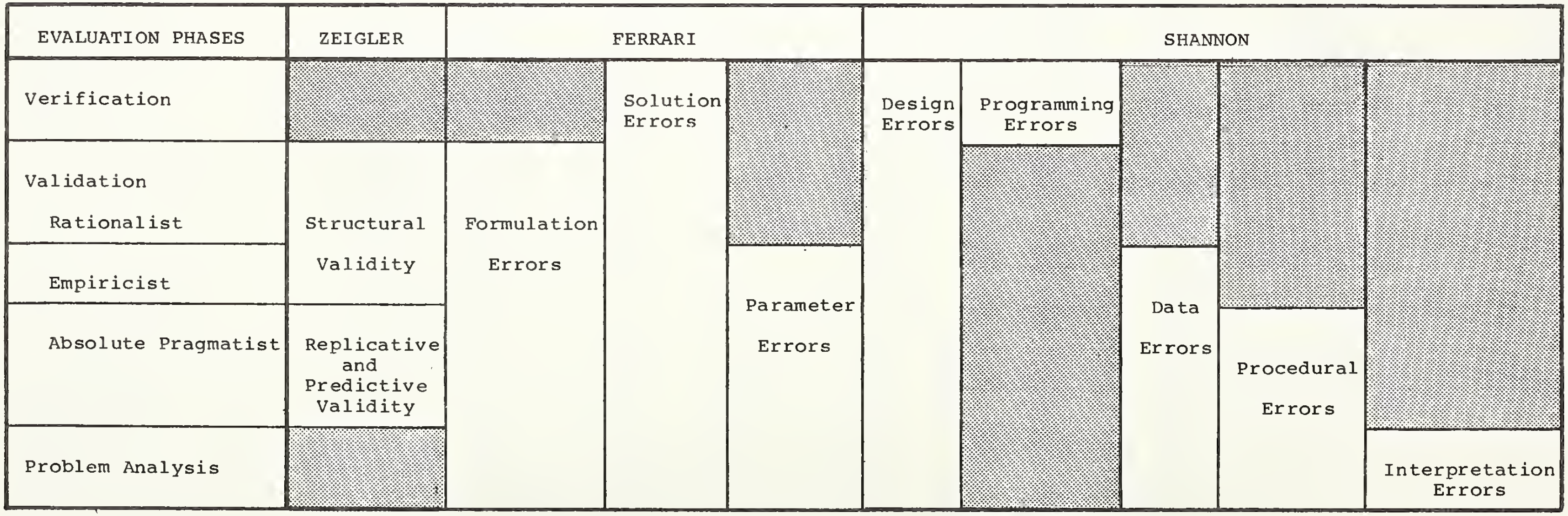


Figure 8.2

providing greater assurance at increased cost to the user. Anshoff and Hayes [ANSH72] suggest that relative costs and benefits resulting from increasing degrees of validation are related in the manner depicted in figure 8.3. This graph, which is also reproduced in Shannon, shows the benefit to cost ratio peaking at something less than perfect validity.

In discussing the maximum tolerable error that can be accepted in a computer system simulation model, Ferrari [FERR78 p137] states:

> In studies which involve comparisons between different systems or between different versions of the same system, what usually matters is not the exact values of performance indices but their sensitivity to the types of changes being considered.

Thus, he argues that relatively low levels of replicative and predictive validity can be tolerated if the model reacts in the same way as the real world system to pertinent changes. Ferrari sees this sensitivity validity as an acceptable but not preferable surrogate for the ideal of predictive validity:

> Having models which also accurately reproduce the values of the (performance) index is certainly sufficient but by no means necessary ...

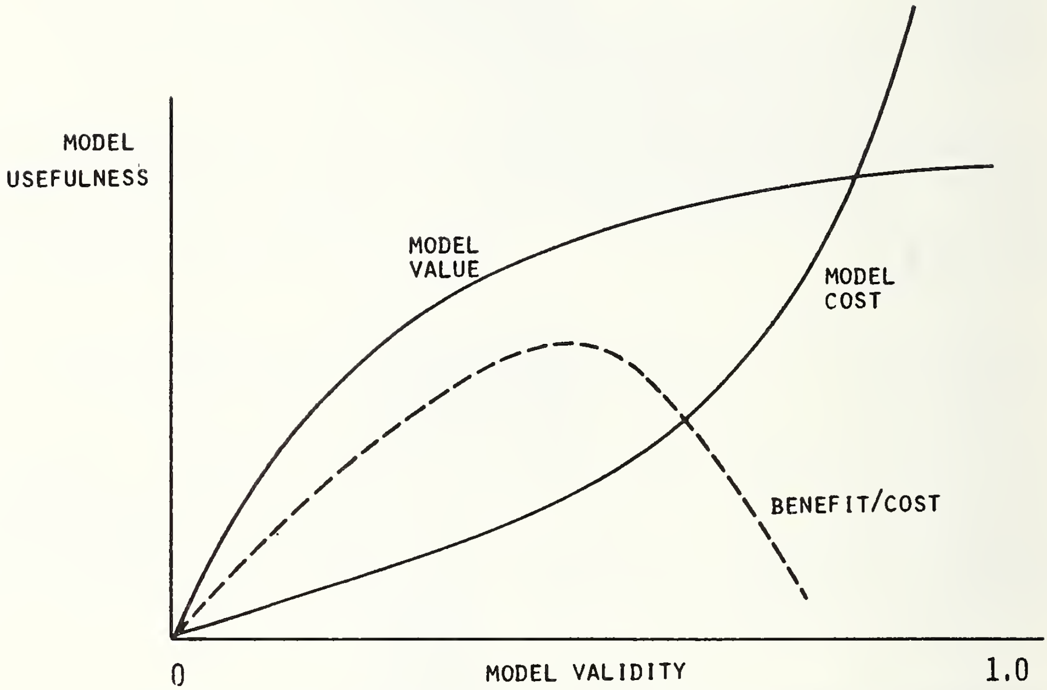
We conclude that, like the evaluation process itself, acceptance criteria are determined based upon the objectives of the modeling effort. Furthermore, these criteria are continually reviewed throughout the model development and implementation project.

8.5 Computer System Model Evaluation

Computer system modeling is unique in that, unlike many real world systems, computers can be measured and often controlled by the modeler. This provides an opportunity for validation that is uncommon with other modeling objects. Seven years ago, Reitman [REIT71 p339] wrote:

> The best examples of verification of simulation predictions with actual experience should come from the computer system designers. They have well documented existing computer systems, input data, and the advantage of having the computer. In spite of these advantages, the verification of simulations of complete computer systems is quite rare.

MODEL VALIDATION COSTS AND BENEFITS



Source: Ansoff, H.I., and R.L. Hayes, "Role of Models in Corporate Decision Making," Proceedings of IFORS Sixth International Conference, Dublin, Ireland, August 1972.

Reprinted in:

Shannon, R.E., Systems Simulation, The Art and Science, Prentice-Hall, Englewood Cliffs, NJ, 1975, p.209.

Figure 8.3

Some progress has been made since this somewhat discouraging assessment. Conferences [HIGH73-76, BEIL77] and books [FERR78] address computer system performance modeling and evaluation issues including model evaluation. Trace driven modeling [SHER72-73, NOE72] was developed in large part to overcome some of the complexities associated with validating stochastic computer system models through the use of event traces rather than random numbers and random variables for determining process sequences. Clearly, more evaluation of computer system models is attempted today. Nevertheless, many computer system models are still not validated experimentally or empirically.

The research described in this document is concerned with database management system modeling. How many DBMS models have been evaluated for accuracy and correctness? Literature reviews and personal discussions with many of those working in this field lead to the conclusion that little has been accomplished in this area. Validation generally follows the philosophy of rationalism; validity is implied from the acceptance of model representations for basic system components and their relationships. Models of database design concepts have rarely been built prior to their implementation. Thus, there is virtually no precedent for the approach taken in the set processor modeling effort described herein. From the outset, the ability to validate model predictions was a major objective in implementing the integrated prototype - measurement system - SPPM components of the set processor performance prediction system. The next section describes SPPM evaluation procedures using the concepts and terminology presented above.

8.6 SPPM Evaluation

Evaluation of the set processor performance model has begun. A process that will continue throughout the model life cycle, SPPM evaluation has as its objective assurance of the strongest forms of isomorphism and predictive validity. As described above, the SPPM actually contains two models: the size estimation modeler, and the response estimation modeler. These two models are built on a foundation of partially overlapping sets of components and shared storage for parameters and derived values. Consequently, verification and validation of SPPM components may contribute to the evaluation of both size and response modelers. Because the size model has been available for a longer period of time, the evaluation process has progressed further for it than for the response modeler. Some progress has been made, however, in the initial evaluation steps for the response model. SPPM evaluation progress and plans are described in the remaining paragraphs in this chapter in terms of the multi-phase utilitarian approach.

8.6.1 Verification. All model components have been reviewed to determine that they faithfully carry out the model design. This does not mean that SPPM subroutines are error free, but rather that a comprehensive debugging effort has been completed. Various techniques were employed to assist and guide the verification task.

* Careful selection of test parameters and query loads to exercise model capabilities fully within time and computer processing resource constraints.

* Use of an interactive debugging facility to monitor execution of model subroutines; because of the complexity of the SPPM, some type of debugging aid is almost essential for finding errors such as inconsistent subroutine arguments in calling and called programs, untested program paths, and logical design and coding errors.

* Review of virtually all SPPM procedures; this was done in order to prepare the pseudo FORTRAN statements appearing in Chapter 7 of this document. A number of previously undetected errors were found through this review. Indeed, verification was probably the most valuable contribution from this tedious and time consuming endeavor.

Beyond these initial verification activities, other evaluation phases have uncovered errors in model program realizations; thus, several iterations of verification have already been completed. Of course, as evaluation continues more verification steps are certain to be required.

8.6.2 Validation. The multi-stage utilitarian validation approach described above is applicable to the SPPM modeling effort. Model design and development has attempted to achieve structural validity; that is, there is isomorphism between model and prototype DBMS operation. This is especially true for the response modeler that uses modified PSP programs to determine the sequence of DBMS functions required to answer a query. Over and above structural validity, an ongoing effort to achieve replicative and predictive validity using all three philosophical stages is underway. While all model components have achieved some degree of validation, the iterative multi-stage evaluation process is expected to continue.

STAGE 1 - Rationalism All SPPM components have rational bases in the operation of the PSP prototype and stand up to careful scrutiny for reasonableness. Other database management system modeling efforts have not gone beyond this step in their validation. Because of the availability of measurement data and the Positional Set Processor prototype, other stages of validation can be considered for the SPPM as well.

STAGE 2 - Empiricism Major SPPM components have been the subject of validation experiments. Using the prototype DBMS and measurement system, modules such as the following have been calibrated against corresponding PSP facilities.

* Model representation of I/Ø buffer manipulation by operating system and (FORTRAN) language software was tested against, and ultimately changed because of, experimental results derived from the PSP on a quiescent system.

* The use of traversal as a surrogate process for classical set operations was tested and confirmed through analysis of PSP measurements and traversal experiments.

* The collection of database functions was confirmed and modified based on accumulated measurements of multiple PSP sessions showing selected functions as significant in relation to both total resource requirements and other functions.

Empirical and/or experimental validation of component processes does not consider their interaction. Consequently, a still stronger validation stage 3 is desired.

STAGE 3 - Absolute pragmatism The true test of the integrated prototype - measurement - modeling approach employed in this research is whether the SPPM can predict the performance of the PSP prototype. We seek first replicative and then predictive validity. When performance indices can not be replicated, we recognize (but are not necessarily satisfied with) the relative sensitivity validity described previously. Current status and future validation plans for the two SPPM models are described in the next chapter.

8.6.3 Problem analysis. The SPPM is just now beginning to be used. Consequently, there have been few opportunities to interpret and use model results. Some initial model results did provide the insight necessary to guide modifications in the PSP prototype that greatly enhanced its usefulness as a research and demonstration tool. The success of these modifications attests to the correctness of the analysis and interpretation of results. Early size model estimates have shown considerably larger portions of secondary storage used for overhead than had been thought, with a resulting greater media space requirement. These and other model results will be carefully considered before implementation and design decisions are made.

8.7 Summary

This chapter has reviewed some of the literature and terminology in the area of model evaluation. A consensus approach comprised of three phases - verification, validation, and problem analysis - was adopted. Validation was seen as an iterative, multi-stage process falling in the middle of the philosophical extremes of rationalism, empiricism and absolute pragmatism. Finally, the current evaluation status of the SPPM size and response models was reviewed; the continuing evaluation process is well underway in both cases, with the size model having already demonstrated a strong form of replicative/predictive validity.

9. RESULTS

9.1 Research Accomplishments

This research has included activities in the following areas.

* Prototype DBMS Development

* Measurement System Development

* Predictive Modeling
    - Model development
    - Model evaluation
    - Model application

* Research Generalization

Accomplishments in each of these areas are summarized in the following sections.

9.2 Prototype DBMS Development

The heart of the integrated evaluation approach that was developed and demonstrated in this project effort is a limited prototype implementation for the proposed DBMS design. This research included transporting the initial Positional Set Processor prototype implementation to the NBS testbed and substantially enhancing its capabilities. Improvements made during this project are outlined in section 4.3.2; together, they significantly increased the usefulness of the prototype DBMS as a research tool.

Database management systems, even in prototype form, are complex and sophisticated software tools; DBMS development and enhancement are, therefore, difficult and challenging endeavors. Because of the uniqueness of the design concepts imbedded in the prototype, much of the PSP software is without precedent. The secondary indexing mechanism that was built using a true set processor is just one example.

In addition to accomplishments that are specific to the Positional Set Processor prototype, this research involved the use of methodologies and tools that can be applied to developing other DBMS prototype software. Objectives and procedures for implementing DBMS prototypes are described in section 3.2.1. Developing and enhancing the PSP also provided insight into the determination of features that should be incorporated into a limited DBMS prototype. These include:

* basic database loading, accessing and updating capabilities;

* a secondary storage utilization strategy (i.e., a representative DBMS can not run entirely in main memory);

* a primitive user interface; and

* features necessary to demonstrate and test design characteristics (e.g., a prototype for a design utilizing abstract data types should include integrity features that might be ignored in other prototypes).

Because of the proprietary nature of DBMS products, it is difficult to find statistics describing development costs for database management software systems. Individuals participating in the early stages of its development estimate that one widely used commercial database management system required over twenty-five (25) man-years of development effort [PERS78]. On the other hand, the approach proposed in this research was predicated on the belief that a limited DBMS prototype can be developed with approximately 10% of the resources and time required for the complete system. Actual personnel time for all prototype software development as well as enhancement under this project is estimated to be about thirty (30) man-months. Because many of the high-level software tools developed for the Positional Set Processor can be used, the development of other DBMS prototypes should require even fewer resources.

While a prototype is not intended as an operational software product, it can provide the foundation for an iterative development process leading to a full-scale implementation. The early availability of the prototype for observation and modeling is, of course, one of its most important characteristics. The integrated design evaluation approach focuses on the questions of whether and how full-scale implementation should be carried out; if the prototype and model predictions indicate that the design is not viable, the cost is only a fraction of that for an abortive complete development effort.

9.3 Measurement System Development

The integrated design evaluation approach calls for the use of a measurement system for observing prototype DBMS performance. Measured results provide insight into prototype operations and are used for calibrating the model and deriving parameters. Measured performance indicators are compared to model predictions for validating the model. The measurement system developed for this project is described in chapter 5; it is both flexible and simple to use. Four characteristics differentiate it from previous measurement efforts.

* Written in a high-level language (FORTRAN), the system is both transportable and easy to understand.

* The system is designed for measuring procedure-level events; that is, subroutine entries and exits are recorded. This contrasts with the machine and systems software dependent approach of measuring operating system service requests.

* Accurate measurements can be obtained even with a coarse and unpredictable system clock facility.

* The object software is viewed as a hierarchical collection of procedures. The flexible user interface allows measurement of specific procedures, groups of procedures, and trees within the software hierarchy.

The measurement and analysis system is sufficiently general to be applicable to other DBMS design evaluation projects with little modification.

9.4 Predictive Modeling

Using the PSP prototype and measured observations of its performance as an object, this research concentrated on developing models for predicting gross indicators of performance for database management systems using PSP design concepts. Few models of DBMS software have been built previously. Past efforts have concentrated on specific DBMS components and/or on designing <u>databases</u> rather than on designing <u>database management systems</u>. No existing models were capable of representing the PSP's compacted bit-string encoding and processing of sets. Accomplishments in three aspects of predictive modeling are discussed in the following paragraphs.

9.4.1 Model development. The Set Processor Performance Model is not just a modeling concept, but has been implemented fully. Described in chapters 5 through 7, the SPPM is made up of two major predictive components: the analytic size estimation modeler, and the stochastic response time simulator. Each of the two modelers represents a separate contribution.

Size modeler The size modeler is a general, analytic tool for predicting secondary storage utilization for a given DBMS design strategy. It incorporates several concepts that differentiate it from other storage structure models.

* It has sufficient power and flexibility to handle the complex and unique PSP table structure.

* At the same time, it is general enough to represent other DBMS storage utilization strategies.

* The elementary file representation framework for describing secondary storage structures is a major improvement over the modeling of Senko and Owens.

* The functional taxonomy for storage structures provides a framework for evaluating secondary storage utilization strategies not previously available.

* Finally, logical and elementary file parameters provide a comprehensive summary of database characteristics that determine secondary storage requirements and influence response characteristics.

Response modeler The response time estimation component of the SPPM is a PSP prototype specific, stochastic model that has both analytic and simulation components. It differs from the few other response time estimation DBMS models in several ways.

* It is tailored to Positional Set Processor design concepts that can not be represented using other modeling tools.

* It is built on a framework of set processor primitives and DBMS model utility functions that can be applied to increasingly broad classes of set processors and database management systems. respectively.

* Intermediate model results can be specified by the user or, optionally, can be estimated by Monte Carlo processes within model determined feasible ranges.

* The model, like the DBMS object, is query driven; the user merely formulates commands in the same syntax as that used by the prototype system.

* The response modeler has a "piggy back" relationship with the size estimation model. Parameters for and results generated by the size modeler are used in the response estimation process.

9.4.2 Model evaluation. From the outset, the SPPM modeling effort has had as an objective the validation of model predictions against measured prototype DBMS performance indicators. There is virtually no database management system modeling precedent for this strong validation orientation. The continuing SPPM evaluation process has as its ultimate objective the strongest forms of isomorphism and predictive validity. Chapter 8 presents a multi-phase utilitarian approach to model evaluation and describes SPPM evaluation activities. Specific SPPM validation achievements are summarized below.

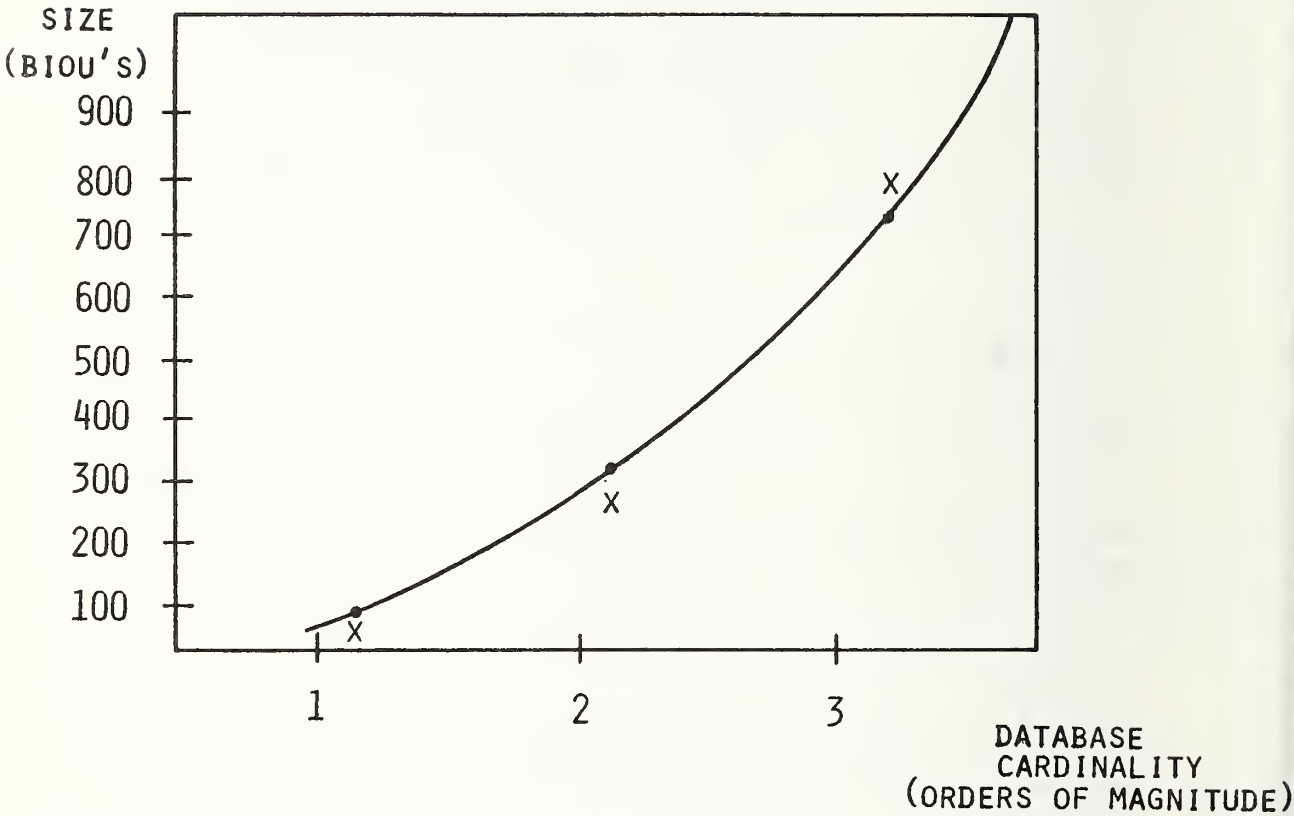
Size modeler status The size modeler is an analytic model that predicts stored database size based on parametric descriptions for database content and logical structure, and secondary storage utilization strategy. Model predictions were matched against the only existing databases for the PSP prototype. Figure 9.1 shows the close relationship between predicted (marked with an X) and actual database secondary storage requirements for the three PSP databases. The prediction error for the three databases ranged between -2% and +.4%, with the error decreasing as a percentage of database size for increasingly large databases.

This high correlation between predicted and actual database sizes indicates a high degree of replicative validity. Furthermore, because actual PSP database sizes were in no way known to the model, the size modeler demonstrates a certain amount of predictive validity. Two additional steps to determine predictive validity are planned:

* size modeler estimates will be derived for new PSP databases before they are actually loaded, and

* the size modeler will be used for database management secondary storage strategies other than that used by the PSP prototype.

SPPM ESTIMATES AND PSP DATABASE SIZES



X = MODEL ESTIMATE

. = ACTUAL NBS ECF SECONDARY STORAGE REQUIREMENT

FIGURE 9.1

Regardless of the outcome of future tests for even stronger predictive validity, the SPPM size modeler has proven itself to be a very accurate predictor of PSP stored database size. The remaining validation questions pertain to the range over which predictions are accurate, and the scope of the model vis-a-vis other database design concepts and implementation strategies.

Response modeler status Response modeler validation has not progressed as far as that for the size model. The primary reason for this lag is the fact that the response modeler is dependent on an operational size estimation facility. By necessity, the size modeler was substantially completed a full six months before the first operational version of the response estimation model. Evaluation activities for the size modeler were then carried out in parallel with response model development. Nevertheless, some determinations pertaining to the validity of the response modeler have already been made and others will result from the continuing evaluation process.

First model predictions of response time were off by a factor of two; that is, actual response time was approximately twice the amount predicted by the SPPM. Analysis of model results along with additional STAGE-2 experiments with the PSP prototype indicated that most of the error could be attributed to the I/Ø estimation module and the manner in which it is invoked. The installation of a new operating system may be responsible for I/O estimation problems encountered in early validation attempts. Work is currently underway to recalibrate the I/Ø module.

While the 5Ø% error rate precludes any claims of strong replicative or predictive validity, some degree of sensitivity validity has been demonstrated. Predicted and actual response times move in the same direction by approximately the same amounts (relative to their respective beginning values) when query loads and database parameters are changed.

Even if predicted response exactly matched actual system times, only replicative validity could currently be claimed. This is because of the manner in which times for database function parameters are being determined. In order to estimate response for a specific query and database, database function times are derived by exercising the PSP prototype with the same query and database. Thus, model inputs may represent prior knowledge of system performance. Future plans call for incrementally relaxing this close relationship between model load and the PSP load used for deriving parameters. Eventually, it is hoped that parameters can be derived by accumulating measured results from a query set representing a typical interactive session that is applied to a wide range of databases. At least three increasingly general parameter extraction phases will be used.

* Parameter extraction using the same query and database as that used for response estimation runs.

* Parameter extraction using several queries of the same command type on the same database as that used for response estimation runs.

* Parameter extraction using several queries containing various command types comprising a representative query set on a typical database.

9.4.3 Model application. Regardless of other contributions resulting from this research, the SPPM must be used to evaluate potential PSP performance. While the evaluations of the prototype are not directly related to the research objectives, they will be important determinants of future implementation strategies for the PSP. Questions of potential performance will be answered by perturbing model parameters and query loads beyond those for the prototype DBMS implementation. Preliminary experiments have been carried out using the SPPM size and response modelers. Predictions representing approximately 120 computer runs are summarized below.

Size modeler projections Preliminary experiments using the size modeler have projected both horizontal and vertical database dimensions several orders of magnitude beyond those actually loaded using the PSP prototype. The results of these initial projections, while not totally unexpected, do provide new insights into PSP secondary storage utilization strategies. Model predictions were generated in the form of source and stored database summary and detailed analysis reports illustrated in Figures 7.2 and 7.3.

Horizontal dimensions (attributes and relations) were perturbed by four orders of magnitude beyond the largest actual PSP database. Stored database size estimates grew proportionally with source database size. These results are graphically illustrated in Figure 9.2.

Size estimates were also derived for vertical (tuple) dimension perturbations covering six orders of magnitude beyond existing PSP databases. The kinked relationship between source database size and estimated database storage requirements is illustrated in Figure 9.3; this is common behavior for a DBMS. Initial growth for stored databases was estimated to be at a much slower rate than increases in source database size. Even after the upward turn in the

PRELIMINARY SIZE PROJECTION

(HORIZONTAL PERTURBATIONS)

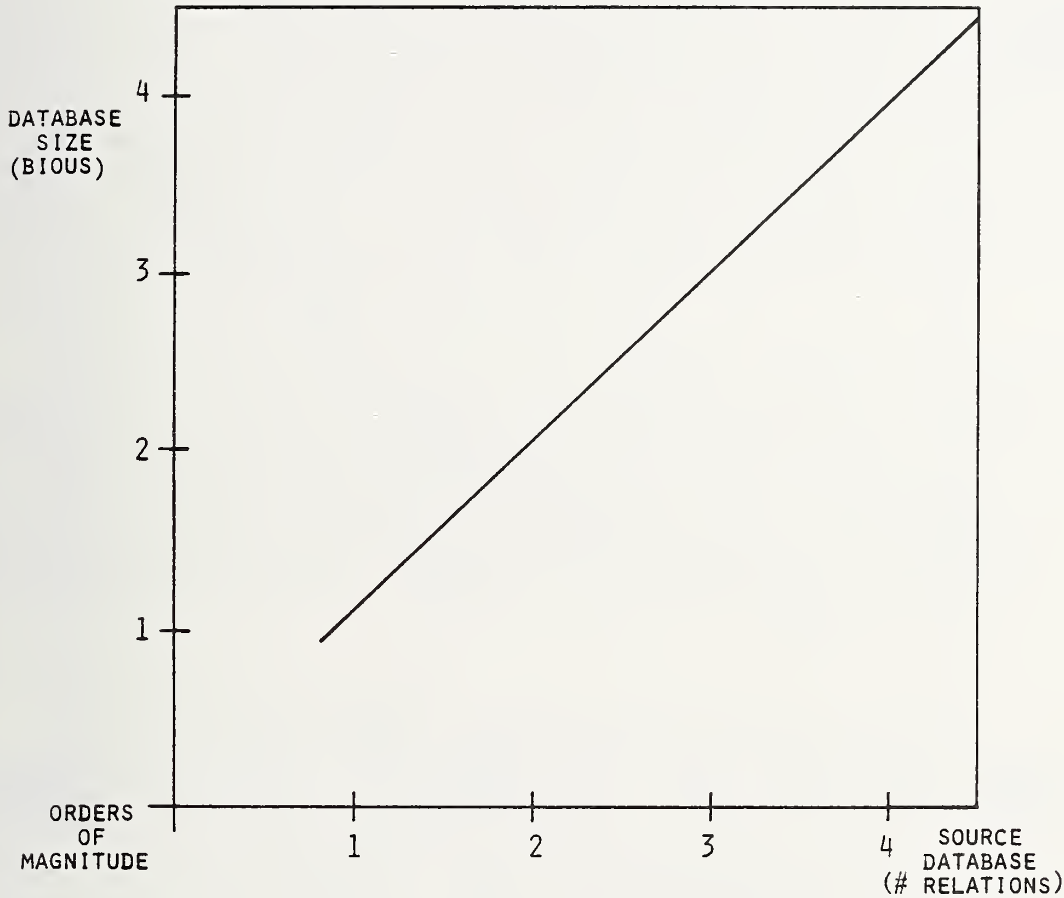


FIGURE 9.2

PRELIMINARY SIZE PROJECTION

(VERTICAL PERTURBATIONS)

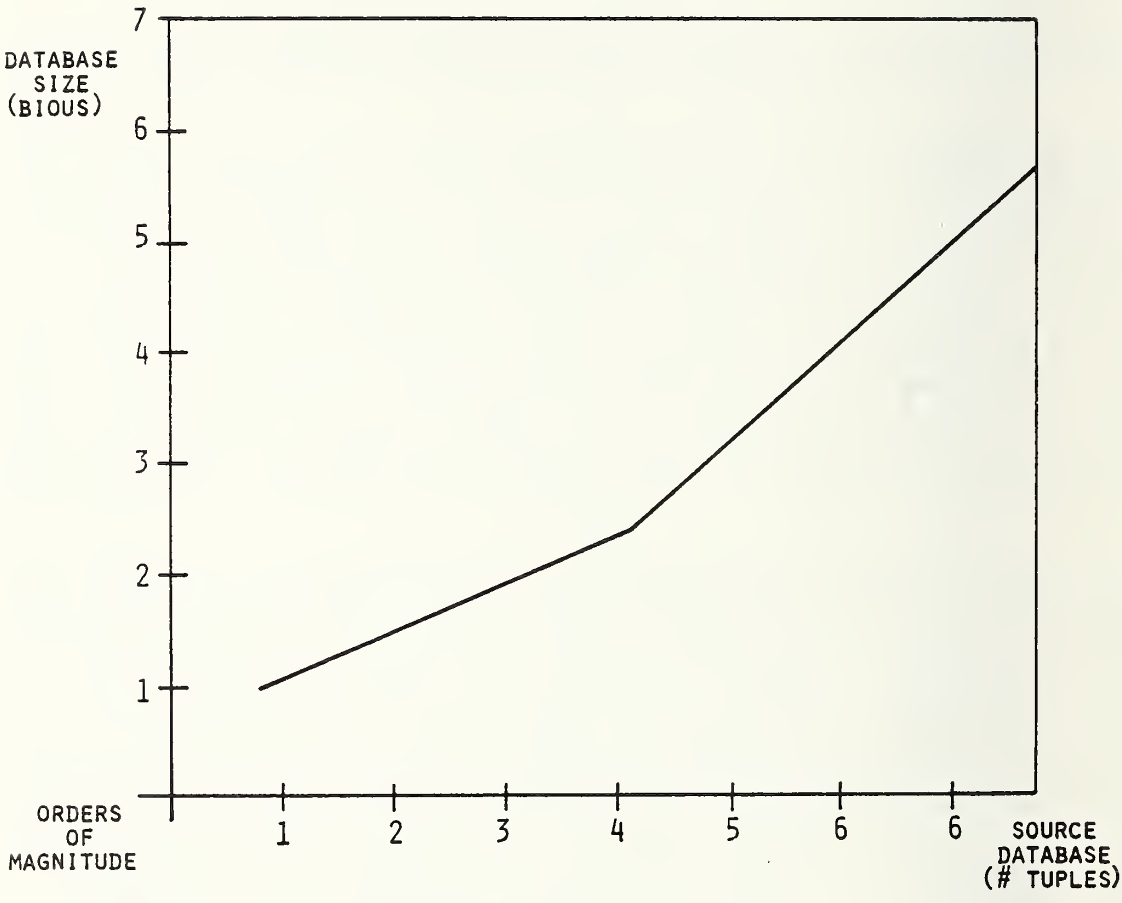


FIGURE 9.3

graph, PSP databases were estimated to grow only directly with source database size. This contrasts with many commercial systems; one widely used DBMS product demonstrated an explosion factor of 7 to 8 times in recent NBS tests.

Response modeler projections Because the response modeler is still in the early stages of evaluation, with only weak sensitivity validity demonstrated, response projections must be viewed as extremely tentative. Nevertheless, response estimates were derived for a single simple equality condition on an indexed attribute applied to three databases of exponentially increasing size. Response estimates were obtained for solution sets with a cardinality of 2, and for solution sets with the maximum cardinalities for the attributes and databases being queried. Figure 9.4 summarizes these tentative results extracted for I/O and response summary and detailed analysis reports illustrated in Figures 7.4 and 7.5. Response time is shown as being related primarily to solution set cardinality. Furthermore, an analysis of estimated times for specific database functions shows a single activity, adding an element to a set, requiring 99% of total time for queries with large solution sets.

These results were not anticipated. Should subsequent experiments confirm these tentative findings, they will be important in determining where technology could be applied to improve prototype performance; for instance, these preliminary findings are strong arguments for considering the use of special purpose hardware to dramatically improve the set building process. The SPPM will be used to further examine this and other potential implementation approaches.

9.5 Generality of Results

The purpose of this research was the development of a DBMS design evaluation methodology. The development of this methodology focused on the design concepts imbedded in the Positional Set Processor prototype implementation. From the PSP evaluation effort, both methodological and software tools were developed; many of these tools can be applied also to other DBMS design evaluation efforts. The following sections discuss these tools and their generality.

PRELIMINARY RESPONSE PROJECTION

(SIMPLE EQUALITY CONDITION ON INDEXED ATTRIBUTE)

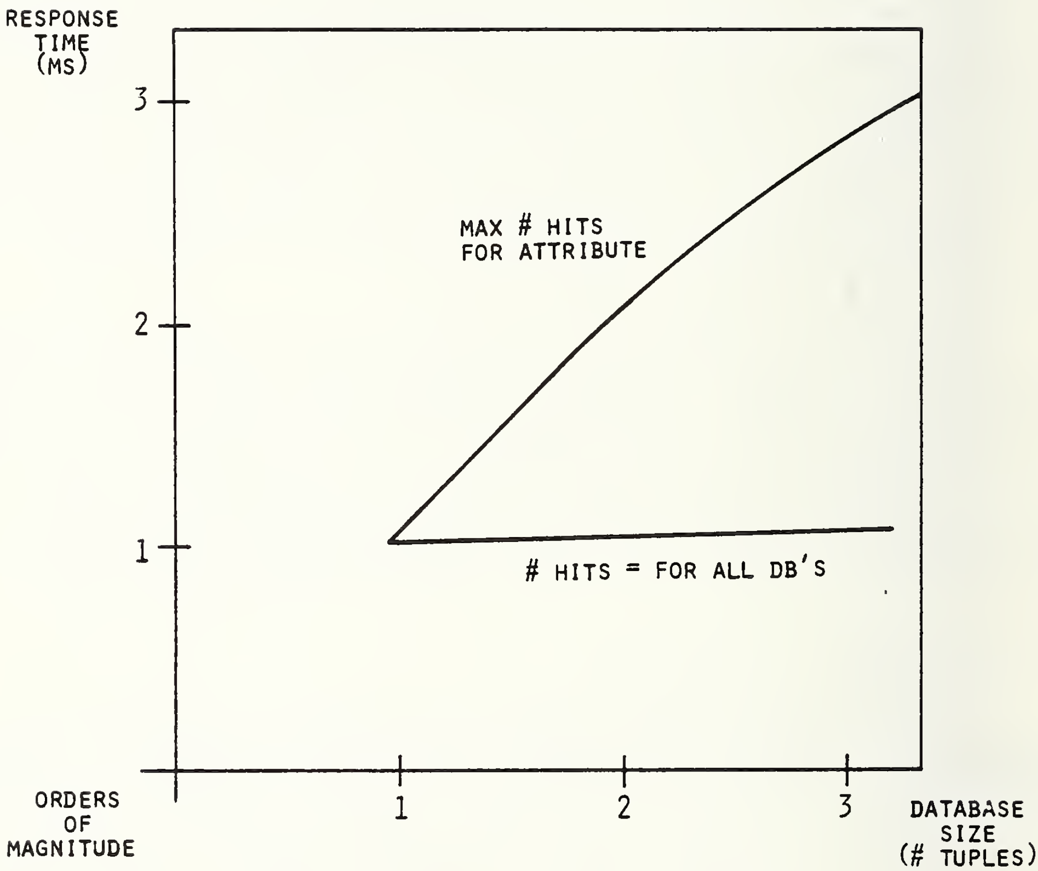


FIGURE 9.4

9.5.1 Evaluation methodology. The integrated approach that is successfully being used to evaluate PSP design concepts, could be applied to other proposed DBMS designs as well. Starting with a set of design specifications for a database management software/hardware facility, the steps below would be followed.

1. Develop, using a high-level language, a well structured prototype incorporating only essential features for the proposed DBMS design. Prototype implementation should follow the guidelines outlined in section 3.2.1.

2. Exercise the prototype to assure the efficacy of the DBMS design approach; that is, use the prototype implementation to prove gross technological feasibility for the underlying design concepts.

3. Instrument the prototype DBMS; that is, insert measurement system "hooks" or probes. For the PSP measurement facility, probes take the form of CALLS to measurement program SVC4ØØ that are inserted in DBMS procedures. Of course, these probes can be inserted at the time prototype subroutines are written. The measurement system developed for this research can be used on any system supporting a FORTRAN compiler. Its use requires that the prototype DBMS be written in a language that has a FORTRAN linkage facility. Part of the instrumentation task is to calibrate the measurement facility to overcome limitations in system provided hardware/software clocks.

4. Develop and use a model based on the SPPM to examine the performance potential of the proposed design concepts. Section 6.7 outlines eleven steps to be followed in an iterative model development and implementation process.

9.5.2 Applicability of software tools. A major portion of the software developed for this research could be used for other DBMS design evaluation projects as well as for database design studies. The applicability of software tools in each of the major performance evaluation system components to other evaluation projects is described below.

Positional set processor prototype The PSP prototype is not generally applicable to other DBMS design evaluation efforts. In developing the Positional Set Processor, however, a number of general utilities were required. A substantial library of FORTRAN subroutines and functions necessary for implementing a DBMS in a high-level language is available. The routines range from bit and character manipulation utilities, to sophisticated generalized programs for performing non-standard direct access I/O. Approximately 10% of the over 200 PSP subroutines are of this general nature that could be applied to other DBMS implementations. Some are even being used at NBS in software other than database management systems.

Measurement and analysis system Virtually the entire measurement facility could be used for evaluating other DBMS designs. As stated above, the prototype must be coded in a language that provides a FORTRAN linkage. In addition, the measurement system must be calibrated and a table of program reference numbers must be prepared in order to use the measurement facility. In a system with a clock that has sufficiently fine granularity, the cycling mechanism may not be required; because of the modular subroutine structure, this facility can be easily deactivated.

Set processor performance model Section 6.7 shows that approximately 90% of the 215 programs in the SPPM would be applicable to other DBMS design evaluation endeavors. The remaining 10% comprise the sequence selector module that must be coded for each object database management system.

10. CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS

10.1 Conclusions

The major proposition considered by this research was whether proposed database management system designs could be evaluated through an integrated prototype DBMS development, measurement and modeling approach. The Set Processor Performance Prediction System demonstrates the efficacy of this approach; that is, the tools and techniques developed in this research do provide a feasible mechanism for evaluating the performance potential of the design concepts embodied in the Positional Set Processor prototype. In the process of developing the performance prediction system, the three supporting propositions stated in the opening chapter were also proved through demonstration.

10.1.1 (Proposition-1)High-level modeling. Both measurement and performance modeler components of the Set Processor Performance Prediction System consider program level events rather than operating system service requests. The resulting high-level performance measurements and predictions are sufficiently detailed and sensitive to changes in parameters describing DBMS software, load and environmental factors. The impact of operating system scheduling and resource allocation procedures on measured DBMS performance is neutralized by exercising the prototype on a quiescent system. High-level measurement and modeling is consistent with the objectives of evaluating gross performance potential for database design concepts, and with the fact that most DBMS are built on an existing operating system foundation.

10.1.2 (Proposition-2)Measurement system. Closely related to Proposition-1 was the claim that a relatively simple and flexible system could be developed to monitor and record performance data for the prototype DBMS implementation. The Set Processor Measurement and Analysis System accomplishes this objective despite the complexities of a coarse and unpredictable system clock facility.

10.1.3 (Proposition-3)Integrating components. The last proposition dealt with using the prototype DBMS, measurement system, and performance model components in an integrated performance evaluation effort. Each of the following uses for the measured prototype data were mentioned in the proposition; all were applied in the set processor evaluation project.

Understanding DBMS operation Measurement system static and dynamic outputs facilitated investigation of the operation of the DBMS prototype at different levels of detail and from various perspectives. A great deal was learned from simply observing PSP prototype performance. As a result of early measurement system outputs, minor changes were made to the prototype that improved its capabilities and usefulness as a research tool. Furthermore, measurement data identified important procedures for modeling purposes, and (through the formatted path analysis) provided easy to follow graphical explanations of prototype procedure interactions.

Deriving parameter values Measurement data provides parameter values for Set Processor Performance Model executions. The flexible capability for specifying programs to be monitored allowed definition of functions at various levels within the "tree" of programs comprising the Positional Set Processor prototype. Furthermore, investigation and calibration of primitive operations such as packet manipulation and set traversal were accomplished using the measurement facility.

Model validation The use of measured prototype performance to validate model predictions is described in the preceding chapter. Model evaluation, defined previously to be an iterative process including validation, is continuing for the SPPM. Progress so far has shown that the tools provided by the performance evaluation system are sufficient to achieve a high degree of correlation between actual and predicted performance indices.

This research has achieved its objective of developing and demonstrating a methodology for evaluating proposed DBMS design concepts. Initial steps have been taken toward using the Set Processor Performance Model for predicting potential performance for the design concepts imbedded in the Positional Set Processor Prototype. Furthermore, both the methodology and a substantial portion of the software developed for this project could be applied to the evaluation of other database management system design ideas.

10.2 Future Research Directions

This project is merely a beginning; building on this research, future work should address tasks in two broad areas:

* application of the Performance Prediction System and Methodology, and

* extension and enhancement of performance prediction system capabilities.

Research tasks in each of these areas are discussed in the following sections.

<u>10.2.1 Application of prediction system.</u> Because the focus of this research was on the development of methodological and software tools for evaluating database designs as opposed to actual performance evaluation results, a great deal remains to be done in applying the performance prediction system. Potential future research activities in this area include the following.

* Continuation of the iterative evaluation process until response modeler predictions achieve a high level of replicative and predictive validity with respect to the Positional Set Processor prototype. This on-going endeavor is essential for certifying the accuracy of predictions that will be generated in subsequent steps.

* Use of the performance prediction system in a vigorous effort to evaluate the performance potential of the PSP design concepts. This is the single most important future activity; if the performance prediction system and methodology are not used, much of the effort will have only pedagogical benefits. In addition to investigating potential performance limits for the prototype design under various database and query loads, the research should consider the impact of implementation changes such as using hardware for selected set processing primitive functions, and applying different buffer management strategies.

* Application of the SPPM size modeler to database designs other than the positional set processor. In particular, the representation of storage strategies for existing DBMS such as System 2000 would provide a test of the claimed generality and allow further validation of the size modeler.

* Application of the methodology and tools developed for this research to other proposed database management system designs. While a piecemeal application of separate components like the measurement system or size modeler should be encouraged, of greater interest would be the use of all pertinent performance prediction tools in an integrated evaluation effort.

10.2.2 Enhancement of prediction tools. As with any research, many compromises were made in order to accomplish the project objectives. These compromises invariably called for simplicity rather than elegance and limited rather than complete features. There are, therefore, a number of improvements that could be made in the performance prediction system. In addition to adding capabilities that were (for the sake of expediency) knowingly omitted, enhancements could address limitations that became evident only during prolonged use of the prediction system. Three future research projects to extend prediction capabilities are listed below.

* Enhance the SPPM response estimation modeler to handle the entire positional set processor command syntax. This would allow modeling complete PSP query sessions.

* Integrate normative analytic solutions for database storage utilization sub-problems into the SPPM size and response modelers. In particular, the work of Severance [SEVE75] should be considered for inclusion in the SPPM.

* Address the impact of multiprogramming and other operating system complexities. These problems were largely ignored in this research. One possible approach would be to use the SPPM as a front-end for modeling tools that emphasize hardware/software resource allocation and utilization aspects of computer system performance. Another possibility would be to provide an interface between the SPPM and database oriented modelers such as those developed by Reiter [REIE76a-b] and Delutis [DELU77].

(Note: * indicates references appearing in the text.)

AGRA75 Agrawala, A. K. and R. M. Bryant, Models of Memory Scheduling, Computer Science Technical Report TR-392, University of Maryland, July 1975, 22 p.

*ANDA72 Anderson, H. A. Jr., and R. G. Sargent, "Bibliography 31: Modeling, Evaluation and Performance Measurements of Time-Sharing Computing Systems," Computing Reviews of the ACM, Vol. 13, No. 12, December 1972, pp. 6Ø3-6Ø8.

*ANDD77 Anderson, Henry D. and P. B. Berra, "Minimum Cost Selection of Secondary Indices for Formatted Files", ACM Transactions on Database Systems, Vol. 2, No. 1, March 1977, pp. 68-9Ø.

*ANSH72 Anshoff, H. I. and R. L. Hayes, "Role of Models in Corporate Decision Making", Proceedings of IFORS Sixth International Conference, Dublin, Ireland, August 1972.

ARIS74 Aris, John B. B., "Quantifying the Costs and Benefits of Computer Projects", Economics of Informatics, A. B. Frielink, Ed., American Elsevier Publishing Company, Inc., New York, 1974, pp. 15-24.

ASH68 Ash, William and Edgar H. Sibley "TRAMP: A Relational Memory With an Associative Base", University of Michigan Technical Report 5, Ann Arbor, May 1968.

*ASTR76 Astrahan, M. M., et. al., "System R: Relational Approach to Database Management", ACM Transactions on Database Systems, Vol. 1, No. 2, June 1976, pp. 97-137.

BACH72 Bachman, Charles W., "The Evolution of Storage Structures" Communications of the ACM, Vol. 15, No. 7, July 1972, pp. 628-634.

*BAKE75 Baker, F.T., "Structured Programming in a Production Programming Environment," Proceedings of the International Conference on Reliable Software, 21-23 April 1975, Los Angeles, pp172-185, IEEE Computer Society, Long Beach, California.

BARD73 Bard, Y., "Experimental Evaluation of System Performance", IBM Systems Journal, No. 3, 1973, pp. 302-314.

*BASK71 Baskett, Forest, III, Mathematical Models of Multi-programmed Computer Systems, doctoral dissertation, Computation Center, University of Texas at Austin, January 1971, 78 p.

*BEIL77 Beilner, H. and E. Gelenbe, Eds., Modeling and Performance Evaluation of Computer Systems, North-Holland, Amsterdam, 1977, 515 p.

BEIT74 Beitz, Henry E., "A Set-theoretic View of Data-Base Representation", ACM:SIGFIDET Workshop on Data Description, Access and Control, May 1974, pp. 477-494.

*BENJ71 Benjamin, Robert I., Control of the Information System Development Cycle, Wiley-Interscience, New York, 1971, 94 p.

*BLAS75 Blaser, A. and H. Schmutz, Database Research: A Survey, Technical Report 75.10.009, Heidelberg Scientific Center, IBM Germany, November 1975, 91pp.

*BOYS75 Boyse, J. W. and D. R.Warn, "A Straightforward Model for Computer Performance Prediction", Computing Surveys of the ACM, Vol. 7, No. 2, June 1975, pp. 73-94.

*BRIC77 Brice, Richard S. and S. Sherman, "An Extension of the Performance of a Database Manager in a Virtual Memory System Using Partially Locked Virtual Buffers", ACM Transactions on Database Systems, Vol. 2, No. 2, June 1977, pp. 196-207.

*BROO75 Brooks, F. P., Jr., The Mythical Man-Month: Essays on Software Engineering, Addison-Wesley Publishing Co., Reading, Mass., 1975, p. 195.

BROW75 Browne, James C., K. M. Chandy et. al., "Hierarchial Techniques for the Development of Realistic Models of Complex Computer Systems", Proceedings of the IEEE, Vol. 63, No. 6, June 1975, pp. 966-975.

*BUZE73 Buzen, J. P., "Computational Algorithms for Closed Queuing Networks with Exponential Servers", Communications of the ACM, Vol. 16, No. 9, September 1973, pp. 527-539.

*BUZE74 Buzen, J. P. and P. Chen, "Optimal Load Balancing in Memory Hierarchies", Information Processing 74, North-Holland, Amsterdam, 1974, pp. 271-275.

*CARD73 Cardenas, Alfonso F., "Evaluation and Selection of File Organization - A Model and System", Communications of the ACM, Vol. 16, No. 9, September 1973, pp. 540-548.

*CARD75 Cardenas, Alfonso F., "Analysis and Performance of Inverted Data Base Structures", Communications of the ACM, Vol. 18, No. 5, May 1975, pp. 253-263.

*CASE72 Casey, R. G., "Allocations of Copies of a File in an Information Network", Proceedings of AFIPS SJCC 1972, Vol. 40, 1972, pp. 617-225.

CATA72 Catania, Salvatore C., "Computer System Models", Computers and Automation, March 1972, pp. 14-16,18.

*CHAM76 Chamberlin, D. D., M. M. Astrahan, et. al., "SEQUEL 2: A Unified Approach to Data Definition, Manipulation, and Control", IBM Journal of Research and Development, November 1976, pp. 560-757.

*CHAN66 Chang, W., "A Queueing Model of a Simple Case of Time-Sharing, IBM Systems Journal, No. 5, 1966, pp. 115-125.

*CHEG69 Cheng, P. S., "Trace-Driven System Modeling", IBM Systems Journal, Vol. 8, No. 4, 1969, pp. 280-289.

*CHEN76a Chen, Peter Pin-shan, "The Entity-Relationship Model--Toward a Unified View of Data", ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, pp. 9-36.

CHEN76b Chen, Peter Pin-shan, and M. Franklin, Eds., Proceedings of the International Symposium on Computer Performance Modeling, Measurement, and Evaluation, sponsored by ACM / SIGMETRICS and IFIP Working Group 7.3, March 1976, Cambridge, Mass., 326 p.

*CHIL68a Childs, D. L., Feasibility of a set-theoretic data structure: a general structure based on a reconstructed definition of relation. IFIP Congress 1968, North Holland, Amsterdam, 1968, pp. 420-430.

*CHIL68b Childs, D. L., Description of a set-theoretic data structure. AFIPS Conf. Proc., Vol. 33, Part 1, AFIPS Press, Montvale, NJ 1968, pp. 557-564.

*CHIL77 Childs, D. L., "Extended Set Theory: A Formalism for the Design, Implementation, and Operation of Information Systems", Volume IV, Current Trends on Programming Methodology, edited by R. T. Yeh, Prentice Hall, expected publication in early 1977.

*CHUR68 Churchman, C. W., Challenge to Reason, McGraw-Hill Book Co., New York, 1968.

CODA62 CODASYL Development Committee, "An Information Algebra", Phase 1 Report - Language Structure Group, Communications of the ACM, Vol. 5, No. 4, April 1962, pp. 190-204.

*CODD70 Codd, E. F., "A Relational Model of Data for Large Shared Data Bases", Communications of the ACM, Vol. 13, No. 6, June 1970, pp. 377-387.

*CODD71 Codd, E. F., "A Data Base Sublanguage Founded on the Relational Calculus", IBM Research Report, RJ 893 (#15716), July 26, 1971, p. 48.

*CODD72a Codd, E. F., "Further Normalization of the Data Base Relational Model", Data Base Systems: Courant Computer Science Symposium 6, Prentice Hall (1972); also published as IBM Research Report, RJ 909 (#15857), August 31, 1971, p. 33.

*CODD72b Codd, E. F., "Relational Completeness of Data Base Sublanguages", Data Base Systems: Courant Computer Science Symposium 6, Prentice Hall (1972); also published as IBM Research Report, RJ 987 (#17Ø41), March 6, 1972, p. 36.

*COFF67 Coffman, E. G., "Studying Multiprogramming Systems With the Queueing Theory", Datamation, Vol. 13, No. 6, June 1967, pp. 47-54.

*COFF68 Coffman, E. G. and L. Kleinrock, "Feedback Queueing Models for Time-Shared Systems", Journal of the ACM, Vol. 15, No. 4, October 1968, pp. 549-576.

DAHL73 Dahle, Ole Johnny and Jo Piene, "Evaluation of Computer Systems Through Simulation", Management Informatics, Vol. 2, No. 6, 1973, pp. 279-283.

*DATE75 Date, C. J., An Introduction to Data-Base Systems, Addison-Wesley, Reading, Mass., 1975.

DEAN72 Dean, James Elliott, Jr. and C. V. Ramamoorthy, A Hueristic Integer Programming Approach to Certain Types of Computer System Design Trade-Off Decisions, Technical Report No. 128, Information Systems Laboratory, Electronics Research Center, The University of Texas at Austin, Austin, Texas, June 15, 1972.

*DELU77 Delutis, Thomas G., A Methodology for the Performance Evaluation of Information Systems, Report to the National Science Foundation under grant no. GN 36622, March 1977, 183 p.

*DENN68 Denning, P. J., "The Working Set Model for Program Behavior", Communication of the ACM, Vol. 11, No. 5, May 1968, pp. 323-333.

*DEJO79 deJong, S. P., Informal seminar at IBM Yorktown Heights, on "Query-by-Exapmly," January 1979.

*EISN76 Eisner, Mark J. and D. G. Severance, "Mathematical Techniques for Efficient Record Segmentation in Large Shared Databases", Journal of the ACM, Vol. 23, No. 4, October 1976, pp. 619-635.

*ELSP72 Elspas, B., K. N. Levitt, et. al., "An Assessment of Techniques for Proving Program Correctness", Computing Surveys of the ACM, June 1977, Vol. 4, No. 2, pp. 97-147.

*FEDS78 FEDSIM, Evaluation of DBMS Modeling Approaches, Report prepared for U.S. Army Computer Systems Command, Ft. Belvoir, Va., Federal Computer Performance Evaluation and Simulation Center, February 1978, 141 p.

*FERN78 Fernandez, E. B., T. Lang, and C. Wood, "Effect of Replacement Algorithms on a Paged Buffer Database System," IBM Journal of Research and Development, Vol. 22, No. 2, March 1978, pp.185-196.

*FERR78 Ferrari, Damenico, Computer Systems Performance Evaluation, Prentice Hall, Englewood Cliffs, N.J., 1978, 554 p.

*FIFE65 Fife, D. W. and J. L. Smith, "Transmission Capacity of Disc Storage Systems with Concurrent Arm Positioning", IEEE Transactions on Electronic Computers, EC-14, 1965, pp.575-582.

*FIFE77 Fife, D. W., Computer Software Management: A Primer for Project Management and Quality Control, NBS Special Publication 500-11, 1977, 58 p.

*FISH67 Fishman, G. S. and P. J. Kiviat, "The Analysis of Simulation-Generated Time Series", Management Science, Vol. 13, No. 7, March 1967.

FORR68 Forrester, Jay W., Principles of Systems, 2nd Preliminary Edition Wright-Allen Press Inc., Cambridge, Mass., 1968.

*FORR69 Forrester, Jay W., Urban Dynamics, MIT Press, Cambridge, Mass., 1969.

*FORR71 Forrester, Jay W., World Dynamics, Wright-Allen Press, Cambridge Mass., 1971.

FREI72 Freiberger, Walter (Ed.) Statistical Computer Performance Evaluation, Academic Press, New York, 1972, p. 514.

FRIE74 Frielink, A. B., Ed., Economics of Informatics (Proceedings of the IBI-ICC International Symposium), Mainz, September 1974, American Elsevier Publishing Company, Inc., New York, p. 469.

*GASS79 Gass, Saul I., Computer Model Documentation: A Review and an Approach, NBS Special Publication 500-39, 1979, 89p.

*GENT73 Gentleman, W. M. and B. A. Wickman, "Timing on Computers", SIGARCH Quarterly Newsletter, Vol. 2, No. 3, October 1973, pp. 20-23.

GHOS74 Ghosh, S. P. and M. E. Senko, "String Path Search Procedures for Data Base Systems", IBM Journal of Research and Development, September, 1974, pp. 408-422.

*GRAH78 Graham, G. S., Guest Ed., "Special Issue: Queueing Network Models of Computer System Performance", Computing Surveys of the ACM, Vol. 10, No. 3, September 1978, pp. 219-362.

*GRIF75 Griffith, W. G., D. Ingerman and C. Price, "A Simulation Model of UNIVAC's DMS-1100--More Than Just a Performance Evaluation Tool", Proceedings of Symposium on the Simulation of Computer Systems, Boulder, Co., ACM/SIGSIM, August 1975, pp. 90-98.

*HALM64 Halmos, Paul R., Naive Set Theory, D. Van Nostrand Co., New York, 1964, 104p., (See Section 24, pp.94-98, "Cardinal Arithmetic").

*HANT76 Hantler, Sidney L. and J. C.King, "An Introduction to Proving the Correctness of Programs", Computing Surveys of the ACM, Vol. 8, No. 3, September 1976, pp. 331-353.

HARD73a Hardgrave, W. T., "The Prospects for Large Capacity Set Support Systems Imbedded Within Generalized Data Management Systems", International Computing Symposium, Davos, Switzerland, September 1973.

HARD73b Hardgrave, W. T., "Using the Canchy/Cantor Diagonal Method to Implement Concepts From Extended Set Theory", ICASE Report, No. 75-23, Hampton, Virginia, November 1973, p. 24.

HARD73c Hardgrave, W. T., "The Prospects for Large Capacity Set Support Systems Embedded Within Generalized Data Management Systems", presented at the International Computing Symposium 1973, Davos, Switzerland, September 1973.

HARD75a Hardgrave, W. T., "A Technique for Implementing a Set Processor", Technical Report No. 3, Department of Information Systems Management, University of Maryland, December 1975, p. 18.

HARD75b Hardgrave, W. T., "Accessing Technical Data Bases Using STDS: A Collection of Scenarios", ICASE Report, No. 75-8, April 1975, p. 84.

HARD75c Hardgrave, W. T., "Set Processing in a Network Environment", ICASE Report, No. 75-7, March 1975.

*HARD75d Hardgrave, W. T. and E. H. Sibley, "Database Research: Some comments on Future Directions", FDT Bulletin of ACM-SIGMOD, Vol. 7, NO. 3-4, 1975, pp44-48.

*HARD76a Hardgrave, W. T., "A Technique for Implementing a Set Processor", Conference on Data: Abstraction Definition and Structure, ACM, N.Y., 1976, pp. 8-94.

*HARD76b Hardgrave, W. T., "Set Processing: A Tool for Data Management", Fifteenth Annual Technical Symposium: Directions and Challenges, June 1976, ACM, N.Y., pp. 71-8Ø.

*HARD77 Hardgrave, W. T., The Relational Model: A Reformulation of Some Mathematical Aspects, IFSM Technical Report No. 25, Department of Information Systems Management, University of Maryland, College Park, June 1977.

*HARD78 Hardgrave, W. T., The Relational Model: A Reformulation of some Mathematical Aspects Using Positional Set Notation, IFSM Technical Report No. 25, Version 2, Department of Information Systems Management, University of Maryland, College Park, March 1978, 39 p.

*HEIN75 Heindel, L. E. and J. T. Roberto, LANG-PAK: An Interactive Language Design System, American Elsevier Pub. Co., N.Y., 1975, 184pp.

*HIGH73 Highland, H. J., Ed., Proceedings of Symposium on Simulation of Computer Systems, National Bureau of Standards, Gaithersburg, Md., June 1973, 288 p.

*HIGH74 Highland, H. J., Ed., Proceedings of Symposium on Simulation of Computer Systems, National Bureau of Standards, Gaithersburg, Md., June 1974, 210 p.

*HIGH75 Highland, H. J., Ed., Proceedings of Symposium on Simulation of Computer Systems, National Bureau of Standards, Boulder, Co., August 1975, 264 p.

*HIGH76 Highland, H. J., Ed., Proceedings of Symposium on Simulation of Computer Systems, National Bureau of Standards, Boulder, Co., August 1976, 222 p.

HELD75 Held, G. D., M. R. Stonebacker, and E. Wong, "INGRES - A Relational Data Base System", NCC, 1975, pp. 409-416.

HSIA70 Hsiao, David and Frank Harary, "A Formal System for Information Retrieval from Files", Communications of the ACM, Vol. 13, No. 2, February 1970, pp. 67-73.

*HUES67 Huesmann, L.R. and R. P. Goldberg, "Evaluating Computer Systems Through Simulation," Computer Journal, Vol. 10, No. 2, August 1967, pp. 150-155.

*IDC78 IDC, Implementation of Data Base Management Systems, A Research Report Prepared for IDC Information Systems Planning Service Clients, International Data Corporation, June 1978, 221 p.

*IHRE67 Ihrer, Fred C., "Computer Performance Projected Through Simulation", Computers and Automation, April 1967, pp. 22-27.

*KATZ67 Katz, J. H., "An Experimental Model of the System/360," Comm. of the ACM, Vol. 10, No. 11, November 1967, pp. 694-702.

*KERN74 Kernighan, B.W. and Plauger, P. J., _The Elements of Programming Style_, McGraw-Hill Book Company, New York, 1974, 147 p.

*KERN76 Kernighan, B. W. and P. J. Plauger, _Software Tools_, Addison-Wesley Publishing Co., Reading, Mass., 1976, 338 p.

KIMB72a Kimbleton, Stephen R., "Performance Evluation - A Structured Approach", _Proceedings AFIPS 1972 Spring Joint Computer Conference_, pp. 411-416.

KIMB72b Kimbleton, Stephen R., "The Role of Computer System Models in Performance Evaluation", _Communications of the ACM_, Vol. 15, No. 7, July 1972, pp. 586-590.

*KLEI64 Kleinrock, L., "Analysis of a Time-Shared Processor," _Naval Research Logistics Quarterly_, No. 11, 1964, pp. 59-73.

*KLEI66 Kleinrock, L., "Sequential Processing Machines (SSPM) Analyzed with a Queueing Theory Model," _Journal of the ACM_, Vol. 13, No. 2, April 1966, pp. 179-193.

*KOSY73 Kosy, D. W., "An Interim Empirical Evaluation of ECSS for Computer System Simulation Development," _Proc. of ACM SIGSIM Symposium on the Simulation of Computer Systems_, June 1973, pp. 79-90.

*KRAS77 Krasny, Mitchell A.,Ed., _Documentation of Computer Programs and Automated Data Systems_, NBS Special Publication 500-15, 1977, 66 p.

KRIE66 Kriebel, Charles H., "Operations Research in the Design of Management Information Systems", _Operations Research in the Design of Management Information Systems_, John F. Peice, Jr., Ed., 1966, pp. 375-390.

*LANG77 Lang, Thomas, C. Wood and J. B. Fernandez, "Database Buffer Pin Virtual Storage Systems", _ACM Transactions on Database Systems_, Vol. 2, No. 4, December 1977, pp. 339-351.

*LIN76 Lin, Chynan Shiun, D. C. P. Smith and J. M. Smith, "The Design of a Rotating Associative Memory for Relational Database Applications", ACM Transactions on Database Systems, Vol. 1, No. 1, March 1976, pp. 53-65.

*LIPS77 Lipsky, L., and J. D. Church, "Applications of a Queueing Network Model for a Computer System," Computing Surveys of the ACM, Vol. 9, No. 3, September 1977, pp. 2Ø5-221.

*LOWE68 Lowe, T. C., "The Influence of Data Base Characteristics and Usage on Direct Access File Organization," J. ACM, Vol. 15, No. 4, Oct. 1968, pp. 535-548.

*LUCA71 Lucas, Henry C., "Performance Evaluation and Monitoring," Computing Surveys of the ACM, Vol. 3, No. 3, September 1971, pp. 79-91.

*LUM71 Lum, V. Y., and H. Ling, "An Optimization Problem on the Selection of Secondary Keys", Proceedings of the 1971 ACM National Conference, Vol. 26, pp. 349-356.

*LUM75 Lum, V. Y., M. E. Senko et. al., "A Cost Oriented Algorithm for Data Set Allocation in Storage Hierarchies", Communications of the ACM, Vol. 18, No. , 1975, pp. 318-322.

LYON74 Lyons, Norman R. A Model for testing storage allocation policies for on-line disks, Graduate school of Business and Public Administration, Cornell University, Itaca, New York, April 1974, 28 p.

*MACD7Ø MacDongal, M. H., "Computer System Simulation: An Introduction," Computing Surveys of the ACM, Vol. 2, No. 3, September 197Ø, pp. 191-2Ø9. (Bibliography)

*MARC76 March, Salvatore T. and D. G. Severance, The Determination of Efficient Record Segmentation and Blocking Factors for Shared Data Files, MISRC-TR-77-Ø4, Report delivered to David Taylor Naval Ship, R&D Center, October 1976, 31 p.

MARC77 March, Salvatore T. and D. G. Severance, "The Determination of Efficient Record Segmentation and Blocking Factors for Shared Data Files", ACM Transactions on Database Systems, Vol. 2, No. 3, September 1977, pp. 279-296.

*MARC78 March, Salvatore T., Jr., Models of Storage Structures and the Design of Database Records Based Upon a User Characterization, doctoral dissertation, Cornell University, May 1978, 290 p.

*MCKI69 McKinney, J. M., "A Survey of Analytical Time-Sharing Models," Computing Survey of ACM, Vol. 1, No. 2, June 1969, pp. 105-116. (Bibliography)

*MILL76 Mills, Harlan, "Software Development," IEEE Transactions on Software Engineering, December 1976, pp. 265-273.

*NAKA75 Nakamura, F., J. Yoshida and H. Kando, "A Simulation Model for Data Base System Performance Evaluation", Proceedings of the 1975 AFIPS NCC, Vol. 44, 1975, pp. 459-463.

*NAYL67 Naylor, T. H., and J. M. Finger, "Verification of Computer Simulation Models", Management Science, Vol. 14, No. 2, October 1967.

*NBS76 NBS, Guidelines for Documentation of Computer Programs and Automated Data Systems, Federal Information Processing Standards Publication 38, 1976 February 15, 55 p.

*NIEL67 Nielsen, Norman R., "The Simulation of Time-Sharing Systems," Comm. of the ACM, Vol. 10, No. 7, July 1967, pp. 397-412.

*NOE72 Noe, J. D. and G. J. Nutt, "Validation of a Trace-Driven CDC 6400 Simulation", Proceedings of 40th Spring Joint Computer Conference, AFIPS, 1972, pp. 749-757.

*OWEN71 Owens, P. J., "Phase II: A Data Base Management Modeling System", IFIP CONGRES 1971, August 1971, pp. TA22-26.

*OZKA77 Ozkarahan, E. A., S. A. Schuster and K. C. Sevcik, "Performance Evaluation of a Relational Associative Processor", ACM Transactions on Database Systems, Vol. 2, No. 2, June 1977, pp. 175-195.

*PERS78 Personal communication with commercial DBMS vendor's software development staff. Proprietary nature of information precludes specific reference.

PROW74 Prowse, P. H., "Efficiency of Logical Design of Data Structures", Economics of Informatics, A. B. Frieluit, Ed., American Elsevier Publishing Co., Inc., N.Y., 1974, pp. 324-329.

*REIE76a Reiter, Allen and B. Finkel, "Simulating a Virtual Data Machine", NTIS Document No. AD-A027 894, May 1976, 55 p.

*REIE76b Reiter, Allen, On Performance Modelling Of Data Base Management Systems - An Inductive Approach, MRC Technical Report #1648, University of Wisconsin, Madison, July 1976, 42 p.

*REIE77a Reiter, Allen, DIMUI - IDMS User Manual, Version 1.2, Technical Report No. 101, Israel Institute of Technology, June 1977.

*REIE77b Reiter, Allen, and E. Sibley, Simulation and Data Administration, IFSM Technical Report No. 22, Information Systems Management Department, University of Maryland, College Park, July 1977, 27 p.

*REIT71 Reitman, Julian, Computer Simulation Applications, Discrete Event Simulation for Synthesis and Analysis of Computer Systems, Wiley Interscience, 1971, 422 p.

ROTH72 Rothnie, James B., Jr., The Design of Generalized Data Management Systems, Dissertation, Civil Engineering, MIT, September 1972.

ROTH74 Rothnie, James B., Jr., "An Approach To Implementing a Relational Data Management System", ACM-SIGFIDET Workshop on Data Description, Access and Control, May 1974, pp. 277-294.

*ROTH75 Rothnie, James B., Jr., "Evaluating Inter-Entry Retrieval Expressions in a Database Management System", Proceedings of AFIPS National Computer Conference, AFIPS Press, Vol. 44, 1975.

*SALT70 Saltzer, Jerome H. and John W. Guitell, "The Instrumentation of Multics", Communications of the ACM, Vol. 13, No. 8, August 1970, pp. 495-500.

*SCHA67 Scherr, A. L., An Analysis of Time-Shared Computer Systems, MIT Press, Cambridge, Mass., 1967.

*SCHE76 Scheuermann, Peter, A Simulation Model for Data Base Systems, Doctoral dissertation, State University of New York at Stony Brook, 1976, 186 p.

*SCHK75 Schkolnick, M., "Secondary Index Optimization", Proceedings of ACM SIGMOD 1975, International Conference on the Management of Data, San Jose, 1975.

*SCHK78 Schkolnick, Mario, "A Survey of Physical Database Design Methodology and Techniques," Proceedings of Fourth International Conference on Very Large Databases, West Berlin, Germany, 1978, pp474-487.

*SCHW70 Schwetman, Herbert D. Jr., A Study of Resource Utilization and Performance Evaluation of Large-Scale Computer Systems, doctoral dissertation, Computation Center, University of Texas at Austin, August 1970, 115 p.

*SEAM69 Seaman, P. H. and R. C. Sancy, "Simulating Operations Systems," IBM Systems Journal, Vol. 8, No. 4, 1969, pp. 264-279.

*SENK67 Senko, M. E., et. al., Formatted File Organization Techniques: Final Report, Submitted to Rome Air Development Center under contract AF 30(602)-4088, IBM Corp., Yorktown Heights, N.Y., May 1967.

*SENK69 Senko M. E., H. R. Meadow, et. al., File Design Handbook: Final Report, Submitted to Rome Air Development Center under contract AF 30602-69-C-0100, IBM Corp., San Jose, CA., November 1969.

*SENK70 Senko, M. E., Formatted File Organization Techniques: Final Report, IBM Research Report, San Jose, CA, March 1970, Sections VI-VII.

*SENK73 Senko, M. E., E. B. Altman, et. al., "Data Structures and Accessing in Data-Base Systems," IBM Systems Journal, Vol. 12, No. 1, 1973, pp30-93.

*SEVE72 Severance, D. G. and A. G. Merten, "Performance Evaluation of File Organizations Through Modeling", Proceedings of ACM National Conference, August 1972, Boston, pp. 1061-1072.

*SEVE74a Severance, D. G., "Identifier Search Mechanisms: A Survey and Generalized Model", Computing Surveys of the ACM, Vol 6., No. 3, September 1974, pp. 175-194.

*SEVE74b Severance, D. G., "The Evaluation of Data Structures in Data Base System Design", Proceedings of 1974 IEEE International Conference, March 1974.

*SEVE75 Severance, D. G., "A Parametric Model of Alternative File Structures", Information Systems, Vol. 1, Pergamon Press, Great Britain, 1975, pp. 51-55.

*SEVE76a Severance, D. G., "Differential Files: Their Application to the Maintenance of Large Databases", ACM Transactions on Database Systems, Vol. 1, No. 3, September 1976, pp. 256-267.

*SEVE76b Severance, D. G. and R. Duhne, "A Practitioners Guide to Addressing Algorithms", Communications of the ACM, Vol. 19, No. 6, June 1976, pp. 314-326.

*SEVE77 Severance, D. G. and J. V. Carlis, "A Practical Approach to Selecting Record Access Paths", Computing Surveys, Vol. 9, No. 4, December 1977, pp. 259-272.

*SHAN75 Shannon, Robert E., Systems Simulation, The Art and Science, Prentice Hall, Englewood Cliffs, N.J., 1975, 387 p.

*SHER72 Sherman, S., F. Baskett III and J. C. Browne, "Trace Driven Modeling and Analysis of CPU Scheduling in a Multiprogramming System", Communications of the ACM, Vol. 15, No. 12, December 1972, pp. 1063-1069.

*SHER73 Sherman, S. W. and J. C. Browne, "Trace-Driven Modeling: Review and Overview", Proc. of the Symposium on the Simulation of Computer Systems, ACM-SIGSIM, N.Y., 1973, pp201-207.

*SHER76 Sherman, Stephen W., "Trace Driven Modeling: An Update," Proc. of ACM SIGSIM Symposium on the Simulation of Computer Systems, August 1976, pp. 87-91.

*SHNE74 Shneiderman, B., "A Model for Optimizing Indexed File Structures", IJCIS 3, 1974, pp. 93-103.

*SHNE78 Shneiderman, Ben Ed., Data Bases: Improving Usability and Effectiveness, Academic Press, 1978.

SIBL73a Sibley, Edgar H. and Robert W. Taylor, "A data definition and mapping language", Communications of the ACM, Vol. 16, No. 12, December 1973, pp. 750-759.

SIBL73b Sibley, Edgar H., and Alan G. Merton, "Implementation of a generalized data base management system within an organization", Management Informatics, Vol. 2, No. 1, 1973, pp. 21-31.

SIBL74a Sibley, Edgar H., "A system specification language", Commercial Language Systems Infotech State of the Art Report 19, Infotech Information, 1974, pp. 475-503.

SIBL74b Sibley, Edgar H., and Jan A. Turner, "Data Base Management: A Framework For Effective Use", Invited paper, The Second Jerusalem Conference on Information Technology, Jerusalem, July 29-August 1, 1974.

SIBL74c Sibley, Edgar H., and Hasan H. Sayani, "Data Element Dictionaries for the Information Systems Interface", Management of Data Elements in Information Processing, First National Symposium, National Bureau of Standards, January 1974, pp. 285-304.

SIBL74d Sibley, Edgar H. The Data Base Future: System Continuity and Networking, IFSM Technical Report No. , Information Systems Management Department, University of Maryland, College Park, December 1974, 46 p.

SIBL76 Sibley, Edgar H., Guest Editor, ACM Computing Surveys, Special Issue on Database Management Systems, Vol. 8, No. 1, March 1976, 151 p.

SIBL77 Sibley, Edgar H., "Standardization and Database Systems", Proceedings of Third International Conference on Very Large Data Bases, Tokyo, Japan, October 1977, pp. 144-155.

SILE76 Siler, Kenneth F., "A Stochastic Evaluation Model for Database Organizations in Data Retrieval Systems", Communications of the ACM, Vol. 19, No. 2, February 1976, pp. 84-95.

SKOL57 Skolem, T., "Two Remarks on Set Theory", Math. Scand., No. 5, 1957, pp. 40-46.

*SMIT66 Smith, J. L., "An Analysis of Time-Sharing Computer Systems Using Markov Models," Proc. AFIPS 1966 SJCC, Vol. 28, Spartan Books, N.Y., pp. 87-95.

*SOCK78 Sockut, G. H., "A Performance Model for Computer Data-base Reorganization Performed Concurrently with Usage," Operations Research, Vol. 26 No. 5, Sept.-Oct. 1978, pp.789-804.

*STON76 Stonebraker, M., E. Wong, et. al., "The Design and Implementation of INGRES", ACM Transactions on Database Systems, Vol. 1, No. 3, September 1976, pp. 189-222.

*STON77 Stonebraker, M., Informal seminar presented at the Institute for Computer Sciences and Technology, National Bureau of Standards, Summer 1977, on "The Construction of Ingres".

SUND73 Sundgren, Bo An Infological Approach to Data Bases, (Urval nr 7), National Central Bureau of Statistics, Sweden, and University of Stockholm, Department of Administrative Information Processing, Stockholm, 1973, 478 p.

*SVOB76 Svobodova, L., Computer Perfromance Measurement and Evaluation Methods: Analysis and Applications, American Elsevier Pub. Co., N.Y., 1976, 146pp.

*WAGN69 Wagner, Harvey M., Principles of Operations Research, with Applications to Managerial Decisions, Prentice-Hall, Englewood Cliffs, N.J., 1969, 1062 p.

*WORT76 Wortman, David B., "A Study of High-Resolution Timing", Correspondence in IEEE Transactions on Software Engineering, June 1976, Vol. SE-2, No. 2, pp. 135-137.

*YAO74 Yao, S. B., Evaluation and Optimization of File Organization through Analytic Modeling, doctoral dissertation, University of Michigan, 1974.

*YAO77 Yao, S. B., "An Attribute Based Model For Database Access Cost Analysis," ACM Transactions of Database Systems, Vol. 12, No. 1, March 1977, pp45-67.

*YOUC64 Youchah, M. I., et. al., "The Data Processing System Simulator (DPSS), Proc. of AFIPS 1964 FJCC, Vol. 26, p2.1, Spartan Books, N.Y. pp.251-276.

YUE75a Yue, P. C. and C. K. Wong, "Storage Cost Considerations in Secondary Index Selection", International Journal of Computer and Information Sciences, Vol. 4, No. 4, 1975, pp. 307-327.

YUE75b Yue, P. C. and C. K. Wong, "Near-Optimal Heuristics for an Assignment Problem in Mass Storage", International Journal of Computer and Information Sciences, Vol. 4, No. 4, 1975, pp. 281-294.

*ZEIG76 Zeigler, Bernard P., Theory of Modeling and Simulation, Wiley-Interscience, 1976, 435 p.

ZLOO75 Zloof, M. M., "Query by Example", Proceedings of AFIPS 1975 NCC, Vol. 44, AFIPS Press, Montvale, N.J., pp. 431-437.

APPENDIX A: POSITIONAL SET PROCESSOR SCRIPT

This appendix contains a script demonstrating the capabilities of the Positional Set Processor prototype DBMS as of April 1979. Line items starting with a dot "." are commands to the TOPS-1Ø operating system on the NBS Experimental Computer Facility's PDP-1Ø. User inputs are preceded by the DBMS prompt "C>" or by a question requiring a response from the user. All other line items are produced either by the prototype or the operating system.

```
.EX/REL @SIMXPM
LINK:   Loading
[LNKXCT PSMAIN Execution]
SETP - POSITIONAL SET PROCESSOR - NBS ECF

ENTER DATA BASE ID AND MODE (NEW/OLD)
VGH5 OL
 177 UNUSED WORDS IN LEXICON
C>
LIST ALL;

WORKSPACE:


ALIAS TABLE NAMES FOR EXTENDED SETS:

     VGH DATA BASE                 7
     CLYDE                         4

C>
TYPE RE VGH DATA BASE;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T   S         F# G   NAME         IAS       Q        ALT       P      ACC

 139 399        3 1  MONGOLIAN     93.55    39.11    3777.33 1913.57  Ø.53
 139 399        3 1  KOREAN AIR    99.37    33.97    3779.17 1913.15 -Ø.53
 139 399        3 1  JAPAN AIR    111.59    35.57    3779.17 1913.15 -Ø.53
 139 399        3 1  AIR FRANCE   111.97    35.75    3771.11 1913.71 -Ø.55
 139 399        3 1  BRITISH      111.19    35.91    3731.91 1917.57  Ø.59
 139 399        3 1  TWA          111.97    35.75    3771.11 1913.71 -Ø.55
 139 399        3 1  TWA          111.19    35.91    3731.91 1917.57  Ø.59
C>
SUBX *CC=VGH DATA BASE(X.(ALL):.NOT.X.ACC.LE.-.53);
C>
LIST WORK;

WORKSPACE:

     VGH DATA BASE                 7
    *CC                            3
```

```
C>
TYPE RE *CC;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T   S         F# G   NAME         IAS       Q        ALT       P      ACC

 139 399        3 1  MONGOLIAN     93.55   39.11    3777.33 1913.57  Ø.53
 139 399        3 1  BRITISH      111.19   35.91    3731.91 1917.57  Ø.59
 139 399        3 1  TWA          111.19   35.91    3731.91 1917.57  Ø.59
C>
SUBX *DD=VGH DATA BASE(X.(NAME,IAS,Q):X.NAME.EQ.TWA.OR.$
X.IAS.LE.111.59.AND.X.Q.NE.35.91);
C>
TYPE RE *DD;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T   S         F# G   NAME         IAS       Q        ALT       P      ACC

                     MONGOLIAN     93.55   39.11
                     KOREAN AIR    99.37   33.97
                     JAPAN AIR    111.59   35.57
                     TWA          111.97   35.75
                     TWA          111.19   35.91
C>
SUBX *EE=VGH DATA BASE(X.(*):.NOT.X.NAME.EQ.TWA.AND..$
NOT.X.NAME.EQ.BRITISH);
C>
SUBX *FF=VGH DATA BASE(X.(*):[.NOT.X.NAME.EQ.TWA].AND..$
NOT.X.NAME.EQ.BRITISH);
C>
TYPE RE *EE;
ENTER FILE NAME FOR DOMAIN DEFINITION
VGHDOM


     VGH DATA BASE

 T   S         F# G   NAME         IAS       Q        ALT       P      ACC

 139 399        3 1  MONGOLIAN     93.55   39.11    3777.33 1913.57  Ø.53
 139 399        3 1  KOREAN AIR    99.37   33.97    3779.17 1913.15 -Ø.53
 139 399        3 1  JAPAN AIR    111.59   35.57    3779.17 1913.15 -Ø.53
 139 399        3 1  AIR FRANCE   111.97   35.75    3771.11 1913.71 -Ø.55
 139 399        3 1  BRITISH      111.19   35.91    3731.91 1917.57  Ø.59
```

```
C>
TYPE RE *FF;
ENTER FILE NAME FOR DOMAIN DEFINITION
VGHDOM


     VGH DATA BASE

 T   S        F# G   NAME         IAS       Q        ALT       P      ACC

 139 399       3 1  MONGOLIAN     93.55    39.11    3777.33 1913.57  Ø.53
 139 399       3 1  KOREAN AIR    99.37    33.97    3779.17 1913.15 -Ø.53
 139 399       3 1  JAPAN AIR    111.59    35.57    3779.17 1913.15 -Ø.53
 139 399       3 1  AIR FRANCE   111.97    35.75    3771.11 1913.71 -Ø.55
C>
LIST ALL;

WORKSPACE:

     VGH DATA BASE                   7
    *CC                              3
    *DD                              5
    *EE                              5
    *FF                              4

ALIAS TABLE NAMES FOR EXTENDED SETS:

     VGH DATA BASE                   7
     CLYDE                           4

C>
FREE *CC;
C>
FREE *DD;
C>
FREE *EE;
C>
SAVE *FF(CLYDE);
```

```
C>
LIST ALL;

WORKSPACE:

     VGH DATA BASE              7
     CC                         3
     DD                         5
     EE                         5
     FF                         4

ALIAS TABLE NAMES FOR EXTENDED SETS:

     VGH DATA BASE              7
     CLYDE                      4

C>
TYPE XSET VGH DATA BASE;


SET        7 ELEMENTS      39 PACKETS       1TH POSITION
DO YOU WANT A LIST OF M,K,L -- YES OR NØ?
YES


.2780000000D+03      K =    22     L =      1
.4670000000D+03      K =    29     L =      1
.5970000000D+03      K =    33     L =      1
.8630000000D+03      K =    40     L =      1
.1178000000D+04      K =    47     L =      1
.1277000000D+04      K =    49     L =      1
.1328000000D+04      K =    50     L =      1
DO YOU WANT TO SEE HEX DUMP OF PACKETS(YES OR NO)?
YES


888888888 8C7944448 244484412 C34241188 211


C>
SUBX *A=VGH DATA BASE(X.(*):X.NAME.EQ.TWA);
C>
SUBX *B=VGH DATA BASE(X.(*):X.IAS.EQ.111.19);
C>
LET *AINB=*A.IN.*B;
C>
LET *AUNB=*A.UN.*B;
C>
LET *AXUNB=*A.XUN.*B;
C>
LET *ADIFB=*A-*B;
```

```
C>
LET *BDIFA=*B-*A;
C>
LIST WORK;

WORKSPACE:

    *ADIFB                        1
    *BDIFA                        1
     DD                           5
     EE                           5
     FF                           4
    *A                            2
    *B                            2
    *AINB                         1
    *AUNB                         3
    *AXUNB                        2
C>
TYPE RE *A;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T   S        F# G   NAME         IAS       Q       ALT       P      ACC

 139 399       3 1  TWA          111.97   35.75   3771.11 1913.71 -Ø.55
 139 399       3 1  TWA          111.19   35.91   3731.91 1917.57  Ø.59
C>
TYPE RE *B;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T   S        F# G   NAME         IAS       Q       ALT       P      ACC

 139 399       3 1  BRITISH      111.19   35.91   3731.91 1917.57  Ø.59
 139 399       3 1  TWA          111.19   35.91   3731.91 1917.57  Ø.59
C>
TYPE RE *AINB;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T   S        F# G   NAME         IAS       Q       ALT       P      ACC

 139 399       3 1  TWA          111.19   35.91   3731.91 1917.57  Ø.59
```

```
C>
TYPE RE *AUNB;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T'  S         F# G   NAME         IAS       Q       ALT      P      ACC

 139 399        3 1  BRITISH      111.19   35.91   3731.91 1917.57  Ø.59
 139 399        3 1  TWA          111.97   35.75   3771.11 1913.71 -Ø.55
 139 399        3 1  TWA          111.19   35.91   3731.91 1917.57  Ø.59
C>
TYPE RE *AXUNB;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T   S         F# G   NAME         IAS       Q       ALT      P      ACC

 139 399        3 1  BRITISH      111.19   35.91   3731.91 1917.57  Ø.59
 139 399        3 1  TWA          111.97   35.75   3771.11 1913.71 -Ø.55
C>
TYPE RE *ADIFB;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T   S         F# G   NAME         IAS       Q       ALT      P      ACC

 139 399        3 1  TWA          111.97   35.75   3771.11 1913.71 -Ø.55
C>
TYPE RE *BDIFA;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T   S         F# G   NAME         IAS       Q       ALT      P      ACC

 139 399        3 1  BRITISH      111.19   35.91   3731.91 1917.57  Ø.59
C>
SUBX *PROJ=VGH DATA BASE(X.(T,IAS,NAME):X.NAME.EQ.TWA);
```

```
C>
TYPE RE *PROJ;
ENTER FILE NAME FOR DOMAIN DEFINITION
DOMAIN


     VGH DATA BASE

 T   S         F# G   NAME         IAS        Q        ALT        P      ACC

 139                 TWA          111.97
 139                 TWA          111.19
C>
LET *AUNPROJ=*A.UN.*PROJ;
C>
TYPE RE *AUNPROJ USING DOMAIN;


     VGH DATA BASE

 T   S         F# G   NAME         IAS        Q        ALT        P      ACC

 139 399        3 1  TWA          111.97   35.75   3771.11 1913.71 -Ø.55
 139 399        3 1  TWA          111.19   35.91   3731.91 1917.57  Ø.59
 139                 TWA          111.97
 139                 TWA          111.19
C>
SUBX *REALSET=VGH DATA BASE(X.(T,S,IAS):X.IAS.EQ.111.19);
C>
TYPE RE *REALSET USING DOMAIN;


     VGH DATA BASE

 T   S         F# G   NAME         IAS        Q        ALT        P      ACC

 139 399                          111.19
C>
LIST ALL;

WORKSPACE:

    *ADIFB                        1
    *BDIFA                        1
    *PROJ                         2
    *AUNPROJ                      4
    *REALSET                      1
    *A                            2
    *B                            2
    *AINB                         1
    *AUNB                         3
    *AXUNB                        2
```

```
ALIAS TABLE NAMES FOR EXTENDED SETS:

     VGH DATA BASE                     7
     CLYDE                             4

C>
DUMP ALIAS INTO ALIAS;
C>
DUMP ELEMENT INTO ELMT;
C>
DUMP INDEX INTO INDEX;
C>
EXIT;
STOP

END OF EXECUTION
CPU TIME: 5:Ø.26         ELAPSED TIME: 14:54.Ø8
EXIT

.TYPE ALIAS.DAT
 ALIAS TABLE


 HEADER: A7HLEN A7FILE A7NEXT A7WPE

              5 VGH57A     22    2Ø


 ENTRIES: NCHS ELPTR TXPTR NAME
     1      13     2     Ø AIRCRAFT TYPE
     2       1     2     Ø T
     3      13     3     Ø SERIAL NUMBER
     4       1     3     Ø S
     5      13     4     Ø FLIGHT NUMBER
     6       2     4     Ø F#
     7      14     5     Ø GUST INDICATOR
     8       1     5     Ø G
     9       4     6     Ø NAME
    1Ø      18     7     Ø INDICATED AIRSPEED
    11       3     7     Ø IAS
    12      15     8     Ø IMPACT PRESSURE
    13       1     8     Ø Q
    14      17     9     Ø PRESSURE ALTITUDE
    15       3     9     Ø ALT
    16      15    1Ø     Ø STATIC PRESSURE
    17       1    1Ø     Ø P
    18      12    11     Ø ACCELERATION
    19       3    11     Ø ACC
    2Ø      13    51     Ø VGH DATA BASE
    21       5    57     Ø CLYDE
```

```
.TYPE ELMT.DAT

     ELEMENT TABLE


 N7HLEN N7FILE N7NEXT N7WPE N7CPE N7PPE

    5   VGH57E    78    20    25    45

 ENT# CAT SBCT NCH TXPT CARD NPAC DOMS        ENTRY

    1  AT  CH    1    0    0    0    0  #
    2  AT  CH   13    0    0    0    0  AIRCRAFT TYPE
    3  AT  CH   13    0    0    0    0  SERIAL NUMBER
    4  AT  CH   13    0    0    0    0  FLIGHT NUMBER
    5  AT  CH   14    0    0    0    0  GUST INDICATOR
    6  AT  CH    4    0    0    0    0  NAME
    7  AT  CH   18    0    0    0    0  INDICATED AIRSPEED
    8  AT  CH   15    0    0    0    0  IMPACT PRESSURE
    9  AT  CH   17    0    0    0    0  PRESSURE ALTITUDE
   10  AT  CH   15    0    0    0    0  STATIC PRESSURE
   11  AT  CH   12    0    0    0    0  ACCELERATION
   12  AT  RL    5    0    0    0    0            139.000000
   13  AT  RL    5    0    0    0    0            399.000000
   14  AT  RL    5    0    0    0    0              3.000000
   15  AT  RL    5    0    0    0    0              1.000000
   16  AT  CH    9    0    0    0    0  MONGOLIAN
   17  AT  RL    5    0    0    0    0             93.550000
   18  AT  RL    5    0    0    0    0             39.110000
   19  AT  RL    5    0    0    0    0           3777.329990
   20  AT  RL    5    0    0    0    0           1913.570010
   21  AT  RL    5    0    0    0    0              0.530000
   22  XS  TU    0    6   10   43    0  8888888888 8E7221A211 1441F98181
                                        2414114218 421
   23  AT  CH   10    0    0    0    0  KOREAN AIR
   24  AT  RL    5    0    0    0    0             99.370000
   25  AT  RL    5    0    0    0    0             33.970000
   26  AT  RL    5    0    0    0    0           3779.170010
   27  AT  RL    5    0    0    0    0           1913.149990
   28  AT  RL    5    0    0    0    0             -0.530000
   29  XS  TU    0   31   10   45    0  8888888888 8F7221A211 1441312414
                                        1D12411182 48441
   30  AT  CH    9    0    0    0    0  JAPAN AIR
   31  AT  RL    5    0    0    0    0            111.590000
   32  AT  RL    5    0    0    0    0             35.570000
   33  XS  TU    0   56   10   42    0  8888888888 8B7221A211 1441711128
                                        8A24188541 18
   34  AT  CH   10    0    0    0    0  AIR FRANCE
   35  AT  RL    5    0    0    0    0            111.970000
   36  AT  RL    5    0    0    0    0             35.750000
   37  AT  RL    5    0    0    0    0           3771.109990
   38  AT  RL    5    0    0    0    0           1913.710010
```

39	AT	RL	5	Ø	Ø	Ø	Ø	-Ø.55ØØØØ
4Ø	XS	TU	Ø	81	1Ø	46	Ø	8888888888 C97221A211 1441B12281 2242C68822 82884
41	AT	CH	7	Ø	Ø	Ø	Ø	BRITISH
42	AT	RL	5	Ø	Ø	Ø	Ø	111.19ØØØØ
43	AT	RL	5	Ø	Ø	Ø	Ø	35.91ØØØØ
44	AT	RL	5	Ø	Ø	Ø	Ø	3731.91ØØØØ
45	AT	RL	5	Ø	Ø	Ø	Ø	1917.57ØØ1Ø
46	AT	RL	5	Ø	Ø	Ø	Ø	Ø.59ØØØØ
47	XS	TU	Ø	1Ø7	1Ø	46	Ø	8888888888 C87221A211 1441E52124 88A142288A 81288
48	AT	CH	3	Ø	Ø	Ø	Ø	TWA
49	XS	TU	Ø	133	1Ø	46	Ø	8888888888 C97221A211 1441381224 2C68822829 84248
5Ø	XS	TU	Ø	159	1Ø	46	Ø	8888888888 C87221A211 1441E1488B 142288481A 81288
51	XS	RE	Ø	185	7	39	Ø	8888888888 C794444824 4484412C34 241188211
52	XS	TU	Ø	21Ø	3	21	Ø	8888888888 84C9818124 1
53	XS	TU	Ø	233	3	23	Ø	8888888888 8631241418 124
54	XS	TU	Ø	256	3	23	Ø	8888888888 8332882188 118
55	XS	TU	Ø	279	3	24	Ø	8888888888 C138122424 1481
56	XS	TU	Ø	3Ø2	3	23	Ø	8888888888 4C14889142 481
57	XS		Ø	325	4	27	Ø	8888888888 8794444824 4484412
58	XS	TU	Ø	348	3	25	Ø	8888888888 C942212812 41481
59	XS	TU	Ø	371	3	25	Ø	8888888888 C84221C148 81481
6Ø	XS	TU	Ø	394	3	24	Ø	8888888888 C862218218 1488
61	AT	RL	5	Ø	Ø	Ø	Ø	1ØØ.ØØØØØØ
62	AT	RL	5	Ø	Ø	Ø	Ø	3779.ØØØØØØ
63	AT	CH	13	Ø	Ø	Ø	Ø	VGH DATA BASE
64	AT	CH	7	Ø	Ø	Ø	Ø	EASTERN
65	AT	RL	5	Ø	Ø	Ø	Ø	Ø.ØØØØØØ
66	AT	RL	5	Ø	Ø	Ø	Ø	3775.ØØØØØØ
67	AT	RL	5	Ø	Ø	Ø	Ø	1915.ØØØØØØ
68	XS	TU	Ø	417	3	23	Ø	8888888888 86C8812418 421
69	XS	TU	Ø	44Ø	3	24	Ø	8888888888 8721248124 8441
7Ø	XS	TU	Ø	463	3	22	Ø	8888888888 8322888541 18
71	XS	TU	Ø	486	3	24	Ø	8888888888 C191222424 8842
72	XS	TU	Ø	5Ø9	3	24	Ø	8888888888 4E42128142 2888
73	XS	TU	Ø	532	3	24	Ø	8888888888 C112424984 2481
74	XS	TU	Ø	555	3	23	Ø	8888888888 4691424812 888
75	AT	RL	5	Ø	Ø	Ø	Ø	4844.87ØØØØ
76	AT	RL	5	Ø	Ø	Ø	Ø	1537.38ØØ1Ø
77	AT	CH	Ø	Ø	Ø	Ø	Ø	

```
.TYPE INDEX.DAT
 INDEX FILE FOR ELEMENT TABLE

 N5HLEN N5FILE N5NENT N5FULL

     6  VGH55E  3Ø8Ø      77

 POSITION POINTER

     19        51
    217         1
    321        34
    339        22
    345        12
    353        16
    417         7
    425        61
    465        61
    585         6
    617        41
    783        54
    881        11
    9Ø5         3
    921        31
   1Ø17         9
   1Ø75        33
   1Ø91        6Ø
   1145        27
   1169        28
   1171        58
   1177         2
   1185        23
   1321        36
   1391        59
   1433         4
   1457        44
```

----

complete table not displayed for brevity

```
   2985        35
   3Ø17        13
   3Ø18        3Ø
```

```
.RUN SIMDMP

ENTER DATA BASE ID
5
RELATION NUMBER
        51
   ATTRIBUTE NUMBERS
            2
            6
            7
            8
           11
STOP

END OF EXECUTION
CPU TIME: Ø.82  ELAPSED TIME: 2.27
EXIT

.RUN PTRDMP

ENTER DATA BASE ID
5
ENTER DOMAIN DEFINITION FILENAME
DOMAIN
WHERE DO YOU WANT THE FILE TO BE DUMPED ?
ENTER (TTY OR FILE)
>>
TTY


  ATTRIBUTE NUMBER
         1

        VALUE      POINTER      OCCURENCE SET
        139.ØØØØ        2        22,   29,   33,   4Ø,
                                 47,   49,   5Ø,    Ø,

  ATTRIBUTE NUMBER
         2

        VALUE      POINTER      OCCURENCE SET
              16     13Ø         22,
        VALUE      POINTER      OCCURENCE SET
              23     642         29,
        VALUE      POINTER      OCCURENCE SET
              3Ø    1154         33,
        VALUE      POINTER      OCCURENCE SET
              34    1538         4Ø,
        VALUE      POINTER      OCCURENCE SET
              41    2Ø5Ø         47,
        VALUE      POINTER      OCCURENCE SET
              48    2562         49,   5Ø,
```

```
ATTRIBUTE NUMBER
      3

        VALUE        POINTER         OCCURENCE SET
         93.5500       258            22,
        VALUE        POINTER         OCCURENCE SET
         99.3700       770            29,
        VALUE        POINTER         OCCURENCE SET
        111.1900      2178            47,  50,
        VALUE        POINTER         OCCURENCE SET
        111.5900      1282            33,
        VALUE        POINTER         OCCURENCE SET
        111.9700      1666            40,  49,


ATTRIBUTE NUMBER
      4

        VALUE        POINTER         OCCURENCE SET
         33.9700       898            29,
        VALUE        POINTER         OCCURENCE SET
         35.5700      1410            33,
        VALUE        POINTER         OCCURENCE SET
         35.7500      1794            40,  49,
        VALUE        POINTER         OCCURENCE SET
         35.9100      2306            47,  50,
        VALUE        POINTER         OCCURENCE SET
         39.1100       386            22,


ATTRIBUTE NUMBER
      5

        VALUE        POINTER         OCCURENCE SET
         -0.5500      1922            40,  49,
        VALUE        POINTER         OCCURENCE SET
         -0.5300      1026            29,  33,
        VALUE        POINTER         OCCURENCE SET
          0.5300       514            22,
        VALUE        POINTER         OCCURENCE SET
          0.5900      2434            47,  50,
STOP

END OF EXECUTION
CPU TIME: 15.79 ELAPSED TIME: 1:33.97
EXIT
```

APPENDIX B: PERFORMANCE MODEL SCRIPT

This appendix contains a script demonstrating the capabilities of the Set Processor Performance Model as of February, 1979. Line items starting with a single or double dot "." are commands to the TOPS-1Ø operating system on the NBS Experimental Computer Facility's PDP-1Ø. User inputs are preceded by prompts of the form X>, where 'X' is a letter denoting the model driver or functional module requesting user input.

```
.RUN PSPMOD

              POSITIONAL SET PROCESSOR

            PERFORMANCE ESTIMATION MODEL

                   - S P P M -



SPPM COMMANDS - SELECT FROM FOLLOWING LETTERS:
    H = HELP, PRINT THIS SUMMARY
    L = LOAD NEW PARAMETER SET FROM DISK FILE
    D = DISPLAY CURRENT PARAMETER SET
    C = CHANGE PARAMETER SET
    S = SAVE CURRENT PARAMETER SET ON DISK
    Z = RUN SIZE ESTIMATION MODELER
    R = RUN RESPONSE TIME ESTIMATION MODELER
    X = EXIT, TERMINATE EXECUTION OF MODEL

M>
C


SPPM - PARAMETER CHANGE FACILITY


THIS PRELIMINARY SPPM IMPLEMENTATION DOES NOT SUPPORT
INTERACTIVE MODIFICATION OF MODEL PARAMETERS.  A
TEMPORARY MECHANISM FOR CHANGING EXISTING PARAMETERS
IS PROVIDED THROUGH THE GENERATION OF A FORMATTED
PARAMETER LISTING STORED ON DISK THAT CAN BE CHANGED
USING AN ON-LINE TEXT EDITOR AND THEN LOADED WITH
A TABULAR INPUT PROCESSOR.  THE PARAMETER LISTING
HAS THE SAME FORMAT AS THAT PRODUCED BY THE SPPM
DISPLAY(D) FACILITY.  THE LISTING GENERATOR AS WELL
AS THE TABULAR INPUT MECHANISM CAN BE INVOKED
FROM WITHIN THIS CHANGE FACILITY.

INDICATE TABULAR LOAD OR DISPLAY:
    D = GENERATE FORMATTED PARAMETER DISPLAY DISK FILE
    T = INPUT TABULAR PARAMETER DISPLAY FROM DISK


*>
T

ENTER NAME FOR PARAMETER FILE DISPLAY  LISTING
ZLFNEW
PARAMETER FILE <==> ZLFNEW.DIS
```

```
M>
D

SPPM - PARAMETER DISPLAY FACILITY


SPECIFY LOCATION FOR FORMATTED PARAMETER DISPLAY:
    O = DISPLAY AT ON-LINE TERMINAL
    P = GENERATE FILE FOR HIGH-SPEED LINE PRINTER


*>
O


       DATABASE LEVEL PARAMETERS


DBNAM = NAME OF DB
DBRDN = REDUNDANCY % OVER ALL RELATIONS
DBNRL = NO. OF RELATION DEFINITIONS IN DB
DBNDM = NO. OF DOMAIN DEFS IN DB
DBNAT = NO. OF ATTRIBUTE DEFINITIONS IN DB
DBNUA = NO. OF UNIQUE ATTRIBUTES IN DB


DBNAM    DBRDN    DBNRL    DBNDM    DBNAT    DBNUA
--------------------------------------------------
ZLOOF       64        4        7        8        8
--------------------------------------------------
```

RELATION LEVEL PARAMETERS

RLNAM = NAME OF RELATION
RLRPL = NO. OF REPLICATIONS FOR RELATION
RLRDN = REDUNDANCY % OVER ALL ATTRIBUTES
RLDEG = DEGREE: NO. OF ATTRIBUTES IN RELATION
RLCRD = CARDINALITY: NO. OF TUPLES IN RELATION

INDEX*	RLNAM	RLRPL	RLRDN	RLDEG	RLCRD
1	EMP	1	45	4	10
2	SALES	1	54	2	12
3	SUPLY	1	50	2	10
4	TYPE	1	52	3	9

ATTRIBUTE PARAMETERS

ATNAM = NAME OF ATTRIBUTE
ATDOM = NAME OF DOMAIN FOR ATTRIBUTE

INDEX*	ATNAM	ATDOM
1	NAME	NAMDM
2	SALRY	SALDM
3	MGR	NAMDM
4	DEPT	DEPDM
5	ITEM	ITMDM
6	SUPPL	SUPDM
7	COLOR	CLRDM
8	SIZE	SIZDM

DOMAIN PARAMETERS

DMNAM = NAME OF DOMAIN
DMNVL = NO. OF VALUES IN DOMAIN
DMAVS = AVG SIZE IN BITS OF DOMAIN INSTANCES

INDEX*	DMNAM	DMNVL	DMAVS
1	NAMDM	11	360
2	SALDM	11	36
3	DEPDM	5	72
4	ITMDM	6	36
5	SUPDM	4	720
6	CLRDM	4	72
7	SIZDM	3	36

MAPPING ATTRIBUTES ONTO RELATIONS

RLNAM	ATNAM	# OF REPL	INDEXED
EMP	NAME	1	Y
EMP	SALRY	1	N
EMP	MGR	1	N
EMP	DEPT	1	Y
SALES	DEPT	1	N
SALES	ITEM	1	N
SUPLY	ITEM	1	Y
SUPLY	SUPPL	1	Y
TYPE	ITEM	1	Y
TYPE	COLOR	1	N
TYPE	SIZE	1	N

GLOBAL FILE PARAMETERS

GFNEF = NO. OF EF DEFINITIONS IN PARAMETER SET
GFNLE = NO. OF LE TYPES DEFINED FOR ALL EF'S IN PARAMETER SET

GFNEF	GFNLE
10	19

ELEMENTARY FILE PARAMETERS

EFNAM = NAME OF ELEMENTARY FILE (EF)
EFLET = NO. OF LOGICAL ENTRY (LE) TYPES IN EF
EFIXZ = FIXED SIZE FOR EF (INCLUDES ALL LE'S AND ALL OVERHEAD)
EFFOH = O.H. IN BITS FOR EF
EFEOH = O.H. IN BITS FOR EACH LE IN EF
EFNOC = NO. OF OCCURENCES OF EF FOR SPECIFIED RFO AND RFQ
EFRFO = RELATIONAL/FILE OCCURENCE (RFO) INDICATOR
EFRFQ = RELATIONAL/FILE QUALIFIER (RFQ)
EFBUF = I/O SOFTWARE BUFFER SIZE IN BIOUS

INDEX*	EFNAM	EFLET	EFIXZ	EFFOH	EFEOH	EFNOC	EFRFO	EFRFQ	EFBUF
1	DMAIN	2	Ø	Ø	Ø	1			5
2	ALIAS	2	Ø	18Ø	Ø	1			5
3	ELMNT	5	Ø	18Ø	Ø	1			5
4	ETNDX	1	11Ø88Ø	216	Ø	1			5
5	TXTBL	2	Ø	18Ø	Ø	1			5
6	SNDXM	1	36864	Ø	Ø	1			2
7	SWORK	2	Ø	36	*FXLB	1			2
8	SINVS	1	Ø	36	*FXLB	1	RLX		2
9	SRAVI	2	Ø	*AVOH	Ø	1	ATX		2
1Ø	TEMEF	1	Ø	Ø	Ø	1			5

PARAMETER FOR EF LOGICAL ENTRIES

LENAM = NAME OF LOGICAL ENTRY (LE)
LEFUN = FUNCTIONAL TYPE FOR LE
LEFRF = ELEMENTARY FILE REFERENCE
LENOC = NO. OF OCCURENCES OF LE FOR SPECIFIED RFO AND RFQ
LERFO = RELATIONAL/FILE OCCURENCE INDICATOR (RFO)
LERFQ = RELATIONAL/FILE QUALIFIER (RFQ)
LESIZ = SIZE IN BITS OF LE
LEOHD = O.H. IN BITS FOR LE OCCURENCE

INDEX*	LENAM	LEFUN	LEFRF	LENOC	LERFO	LERFQ	LESIZ	LEOHD
1	ATDEF	DEF	DMAIN	1	ATR		576	Ø
2	REDEF	DEF	DMAIN	1	RL		576	Ø
3	ATALI	DEF	ALIAS	2	ATD		72Ø	Ø
4	RLALI	DEF	ALIAS	1	RL		72Ø	Ø
5	SETPI	PRI	ELMNT	1			72Ø	Ø
6	ATPID	PRI	ELMNT	1	ATD		72Ø	Ø
7	ATOM	INS	ELMNT	1	AID		72Ø	Ø
8	TUPLE	PRI	ELMNT	1	TU		72Ø	Ø
9	RELAT	PRI	ELMNT	1	RL		72Ø	Ø
1Ø	EHASH	SEC	ETNDX	1	LE	ELMNT	36	Ø
11	SPNTR	SEC	SNDXM	1	RLX		46Ø8	36
12	UNVRS	SEC	SWORK	1	RLX		*IRBS	54Ø
13	WRKLE	TMP	SWORK	1			Ø	Ø
14	AVSET	SEC	SINVS	1	AIXA		*IIBS	54Ø
15	RAVAL	SEC	SRAVI	1	AIXA		36	Ø
16	RAPTR	SEC	SRAVI	1	AIXA		36	Ø
17	RLBST	PRI	TXTBL	1	RL		*PBST	72Ø
18	TUBST	PRI	TXTBL	1	TU		*PBST	72Ø
19	TEMLE	OVH	TEMEF	1			Ø	Ø

HARDWARE/SOFTWARE ENVIRONMENT PARAMETERS

ENPPI = PROCESSOR POWER INDICATOR: 1 = NBS ECF PDP/1Ø
ENSAU = SIZE IN BITS OF SMALLEST ADDRESSABLE UNIT (SAU)
ENCPS = NUMBER OF CHARACTERS PER SAU
ENBIU = SIZE IN SAU'S OF BASIC I/O UNIT (BIOU)
ENIOM = MAX NO. OF BIOU'S THAT CAN BE TRANSFERRED WITH A SINGLE ACCESS
ENIRA = ACCESS TIME IN MILLISECS FOR SECONDARY STORAGE READ
ENIWA = ACCESS TIME IN MILLISECS FOR SECONDARY STORAGE WRITE
ENIRT = TRANSFER TIME IN MILLISECS FOR SECONDARY STORAGE READ
ENIWT = TRANSFER TIME IN MILLISECS FOR SECONDARY STORAGE WRITE
ENIOP = TIME IN MILLISECS TO OPEN SECONDARY STORAGE FILE
ENICL = TIME IN MILLISECS TO CLOSE SECONDARY STORAGE FILE
ENIDE = TIME IN MILLISECS TO DELETE SECONDARY STORAGE FILE

ENPPI	ENLOD	ENSAU	ENCPS	ENBIU	ENIOM	ENIRA	ENIWA	ENIRT	ENIWT	ENIOP	ENICL	ENIDE
1.ØØ	1.ØØ	36	5	128	5	23	9Ø	51	1Ø4	25	25	371Ø

GLOBAL SOFTWARE PARAMETERS

GSNFN = NO. OF DBMS FUNCTIONS DEFINED

GSNFN
33

SOFTWARE FUNCTION PARAMETERS

FNNAM = NAME OF DBMS FUNCTION
FNPRC = MILLISECS OF PROCESS TIME FOR EACH EXECUTION OF FUNCTION
FNMOD = MODIFICATION FACTOR FOR DBMS FUNCTION PROCESSOR TIME

INDEX	FNNAM	FNPRC	FNMOD
1	M9COM	10	0.00
2	S9PRO	10	0.00
3	M9PID	10	0.00
4	M9ISO	10	0.00
5	M9GTL	10	0.00
6	M9GTS	10	0.00
7	M9GTR	10	0.00
8	A7TRA	10	0.00
9	A7SRC	10	0.00
10	ALLOC	10	0.00
11	DALOC	10	0.00
12	SSAVE	10	0.00
13	SDEST	10	0.00
14	UNION	10	0.00
15	INTRS	10	0.00
16	XUNSD	10	0.00
17	RLCMP	10	0.00
18	SCOPY	10	0.00
19	RGSTR	10	0.00
20	SADD1	10	0.00
21	S9PTR	10	0.00
22	M9ALO	10	0.00
23	S4CHK	10	0.00
24	S4SUB	10	0.00
25	S4EVA	10	0.00
26	S4MOV	10	0.00
27	S4SRC	10	0.00
28	S4IDX	10	0.00
29	S4BLD	10	0.00
30	S4OPR	10	0.00
31	GPTPK	10	0.00
32	TRVRS	10	0.00
33	K9IO	10	0.00

BIT STRING PROCESSING PARAMETERS

BSPKS = NO. OF BITS IN QUATREE PACKET
BSQLV = NO. OF QUATREE LEVELS
BSMPI = MAX NO. OF POSITION ID'S FOR FILE STRUCTURE
BSYNT = Y INTERCEPT FOR TRAVERSAL ESTIMATION
BSX1C = X1 COEF FOR TRAVERSAL SET CARDINALITY
BSX2C = X2 COEF FOR TRAVERSAL SET RANGE

BSPKS	BSQLV	BSMPI	BSYNT	BSX1C	BSX2C
4	16	9			

M>
Z

SPPM - SIZE ESTIMATION MODEL

CALCULATING STORAGE REQUIREMENTS FOR DB ZLOOF
WITH GROSS PARAMETERS:
* 4 DEFINED RELATIONS
* 7 DOMAIN DEFINITIONS
* 8 ATTRIBUTE DEFINITIONS
* 10 ELEMENTARY FILES DEFINED
* 19 LOGICAL ENTRY TYPES DEFINED

ENTER FILE NAME FOR SIZE ESTIMATION REPORT
ZLFNEW
OUTPUT WILL APPEAR IN FILE - ZLFNEW.SIZ

SOURCE DATABASE

ENTITIES	DEFINED	TOTAL DB
RELATIONS	4	4
ATTRIBUTES	11	11
TUPLES	41.000	41.000
SIZE(BITS)	18432.	18432.

```
             STORED DATABASE FILES
----------------------------------------
ENTITIES        DEFINED        TOTAL DB
----------------------------------------
ELEM FILES        1Ø              16
LOG ENTRIES       19           369.ØØ
----------------------------------------
SIZE(BITS)     .34711E+Ø6     .48Ø78E+Ø6
----------------------------------------


              STORAGE UTILIZATION
----------------------------------------
STORAGE FUNCTION               SIZE(BITS)
----------------------------------------
PRIMARY RELATIONSHIPS          43487.
SECONDARY RELATIONSHIPS        22Ø26.
DEFINITION                     23Ø4Ø.
DATA INSTANCES                 288ØØ.
FILE OVERHEAD                  .36343E+Ø6
----------------------------------------
TOTAL STORED DATABASE          .48Ø78E+Ø6
----------------------------------------

M>
H


SPPM COMMANDS - SELECT FROM FOLLOWING LETTERS:
    H = HELP, PRINT THIS SUMMARY
    L = LOAD NEW PARAMETER SET FROM DISK FILE
    D = DISPLAY CURRENT`PARAMETER SET
    C = CHANGE PARAMETER SET
    S = SAVE CURRENT PARAMETER SET ON DISK
    Z = RUN SIZE ESTIMATION MODELER
    R = RUN RESPONSE TIME ESTIMATION MODELER
    X = EXIT, TERMINATE EXECUTION OF MODEL

M>
S


SPPM - PARAMETER STORAGE FACILITY



ENTER NAME FOR PARAMETER FILE
ZLFNEW
PARAMETER FILE <==> ZLFNEW.PAR
```

```
M>
R


SPPM - RESPONSE TIME ESTIMATION MODELER


GROSS PARAMETERS FOR MODELING QUERIES ON DB ZLOOF:
*    4 DEFINED RELATIONS
*    7 DOMAIN DEFINITIONS
*    8 ATTRIBUTE DEFINITIONS
*   1Ø ELEMENTARY FILES DEFINED
*   19 LOGICAL ENTRY TYPES DEFINED
*   33 DBMS FUNCTIONS DEFINED
*   1.ØØ SYSTEM LOAD FACTOR
*   1.ØØ PROCESSOR POWER INDICATOR


ENTER FILE NAME FOR RESPONSE TIME ESTIMATION REPORT
ZLFNEW
OUTPUT WILL APPEAR IN FILE - ZLFNEW.RES



READY TO ACCEPT SPP QUERIES FOR ZLOOF RELATIONS:

    RELATION NAME  NO ATTS  NO TUPLES
    ---------------------------------
        EMP              4        1Ø
        SALES            2        12
        SUPLY            2        1Ø
        TYPE             3         9

USE SPP QUERY FORMAT: EXIT RETURNS TO MODEL DRIVER


C>
SUBX *A=EMP(X.(*):X.NAME.EQ.JONES);

S9PTRB MODELING SUBX PREDICATE EVALUATION:
    ENTER INTEGER NUMBER OF         1Ø TUPLES IN RELATION EMP
    THAT SATISFY EACH ELEMENTARY CONDITION, OR
    ENTER  "?"  FOR MODEL DETERMINATION
**  1-ST COND: NAME  .EQ. JONES...

RGSTR MODELING REGISTRATION OF ATOM:
**IS STRING JONES...IN DB?  ENTER (Y OR N).

:>
Y
ENTER NO IN RANGE     Ø TO     1 , OR ENTER "?" >
1
```

RESPONSE SUMMARY

DESC	QUERY MS	QUERY %	SESSION MS	SESSION %
I/O	981.ØØ	56.35	981.ØØ	56.35
PROCESSING	76Ø.ØØ	43.65	76Ø.ØØ	43.65
OVERHEAD	Ø.ØØØØØE+ØØ	Ø.ØØ	Ø.ØØØØØE+ØØ	Ø.ØØ
RESPONSE	1741.Ø	1ØØ.ØØ	1741.Ø	1ØØ.ØØ

I/O SUMMARY

DESC	QUERY	SESSION
NO PHYSICAL READS	5	5
NO PHYSICAL WRITES	Ø	Ø
NO OTHER I/O'S		
OPEN	Ø	Ø
CLOSE	Ø	Ø
DELETE	Ø	Ø
NO BIOU'S TRANS	16	16
ACCESS TIME	115.ØØ / 11.72%	115.ØØ / 11.72%
TRANSFER TIME	816.ØØ / 83.18%	816.ØØ / 83.18%
OTHER I/O TIME	5Ø.ØØ / 5.1Ø%	5Ø.ØØ / 5.1Ø%
TOTAL TIME	981.ØØ	981.ØØ

```
C>
SUBX *B=SUPLY(X.(*):X.ITEM.GT.2ØØ);

S9PTRB MODELING SUBX PREDICATE EVALUATION:
    ENTER INTEGER NUMBER OF         1Ø TUPLES IN RELATION SUPLY
    THAT SATISFY EACH ELEMENTARY CONDITION, OR
    ENTER  "?"  FOR MODEL DETERMINATION
**  1-ST COND: ITEM  .GT.  2ØØ.Ø

RGSTR MODELING REGISTRATION OF ATOM:
**IS VALUE  2ØØ.Ø    IN DB?  ENTER (Y OR N).

:>
Y
```

ENTER NO IN RANGE Ø TO 1Ø , OR ENTER "?" >
7

RESPONSE SUMMARY

DESC	QUERY		SESSION	
	MS	%	MS	%
I/O	Ø.ØØØØØE+ØØ	Ø.ØØ	981.ØØ	6.45
PROCESSING	1348Ø.	1ØØ.ØØ	1424Ø.	93.55
OVERHEAD	Ø.ØØØØØE+ØØ	Ø.ØØ	Ø.ØØØØØE+ØØ	Ø.ØØ
RESPONSE	1348Ø.	1ØØ.ØØ	15221.	1ØØ.ØØ

I/O SUMMARY

DESC	QUERY	SESSION
NO PHYSICAL READS	Ø	5
NO PHYSICAL WRITES	Ø	Ø
NO OTHER I/O'S		
OPEN	Ø	Ø
CLOSE	Ø	Ø
DELETE	Ø	Ø
NO BIOU'S TRANS	Ø	16
ACCESS TIME	Ø.ØØ / Ø.ØØ%	115.ØØ / 11.72%
TRANSFER TIME	Ø.ØØ / Ø.ØØ%	816.ØØ / 83.18%
OTHER I/O TIME	Ø.ØØ / Ø.ØØ%	5Ø.ØØ / 5.1Ø%
TOTAL TIME	Ø.ØØ	981.ØØ

C>
EXIT;

```
M>
H


SPPM COMMANDS - SELECT FROM FOLLOWING LETTERS:
    H = HELP, PRINT THIS SUMMARY
    L = LOAD NEW PARAMETER SET FROM DISK FILE
    D = DISPLAY CURRENT PARAMETER SET
    C = CHANGE PARAMETER SET
    S = SAVE CURRENT PARAMETER SET ON DISK
    Z = RUN SIZE ESTIMATION MODELER
    R = RUN RESPONSE TIME ESTIMATION MODELER
    X = EXIT, TERMINATE EXECUTION OF MODEL

M>
L


SPPM - PARAMETER LOAD FACILITY



ENTER NAME FOR PARAMETER FILE
VB5NEW
PARAMETER FILE <==> VB5NEW.PAR


M>
D


SPPM - PARAMETER DISPLAY FACILITY



SPECIFY LOCATION FOR FORMATTED PARAMETER DISPLAY:
    O = DISPLAY AT ON-LINE TERMINAL
    P = GENERATE FILE FOR HIGH-SPEED LINE PRINTER


*>
P

ENTER FILE NAME FOR PARAMETER DISPLAY  REPORT
VB5NEW
OUTPUT WILL APPEAR IN FILE - VB5NEW.DIS
```

M>
D

SPPM - PARAMETER DISPLAY FACILITY

SPECIFY LOCATION FOR FORMATTED PARAMETER DISPLAY:
O = DISPLAY AT ON-LINE TERMINAL
P = GENERATE FILE FOR HIGH-SPEED LINE PRINTER

*>
O

DATABASE LEVEL PARAMETERS

DBNAM = NAME OF DB
DBRDN = REDUNDANCY % OVER ALL RELATIONS
DBNRL = NO. OF RELATION DEFINITIONS IN DB
DBNDM = NO. OF DOMAIN DEFS IN DB
DBNAT = NO. OF ATTRIBUTE DEFINITIONS IN DB
DBNUA = NO. OF UNIQUE ATTRIBUTES IN DB

DBNAM	DBRDN	DBNRL	DBNDM	DBNAT	DBNUA
VGH5	54	1	10	10	10

RELATION LEVEL PARAMETERS

RLNAM = NAME OF RELATION
RLRPL = NO. OF REPLICATIONS FOR RELATION
RLRDN = REDUNDANCY % OVER ALL ATTRIBUTES
RLDEG = DEGREE: NO. OF ATTRIBUTES IN RELATION
RLCRD = CARDINALITY: NO. OF TUPLES IN RELATION

INDEX*	RLNAM	RLRPL	RLRDN	RLDEG	RLCRD
1	REL5	1	54	10	7

ATTRIBUTE PARAMETERS

ATNAM = NAME OF ATTRIBUTE
ATDOM = NAME OF DOMAIN FOR ATTRIBUTE

INDEX*	ATNAM	ATDOM
1	T	TDM
2	S	SDM
3	F#	F#DM
4	G	GDM
5	NAME	NAMDM
6	IAS	IASDM
7	Q	QDM
8	ALT	ALTDM
9	P	PDM
1Ø	ACC	ACCDM

DOMAIN PARAMETERS

DMNAM = NAME OF DOMAIN
DMNVL = NO. OF VALUES IN DOMAIN
DMAVS = AVG SIZE IN BITS OF DOMAIN INSTANCES

INDEX*	DMNAM	DMNVL	DMAVS
1	TDM	1	36
2	SDM	1	36
3	F#DM	1	36
4	GDM	1	36
5	NAMDM	6	72
6	IASDM	5	36
7	QDM	5	36
8	ALTDM	4	36
9	PDM	4	36
1Ø	ACCDM	4	36

MAPPING ATTRIBUTES ONTO RELATIONS

RLNAM	ATNAM	# OF REPL	INDEXED
REL5	T	1	Y
REL5	S	1	N
REL5	F#	1	N
REL5	G	1	N
REL5	NAME	1	Y
REL5	IAS	1	Y
REL5	Q	1	Y
REL5	ALT	1	N
REL5	P	1	N
REL5	ACC	1	Y

GLOBAL FILE PARAMETERS

GFNEF = NO. OF EF DEFINITIONS IN PARAMETER SET
GFNLE = NO. OF LE TYPES DEFINED FOR ALL EF'S IN PARAMETER SET

GFNEF	GFNLE
1Ø	19

ELEMENTARY FILE PARAMETERS

EFNAM = NAME OF ELEMENTARY FILE (EF)
EFLET = NO. OF LOGICAL ENTRY (LE) TYPES IN EF
EFIXZ = FIXED SIZE FOR EF (INCLUDES ALL LE'S AND ALL OVERHEAD)
EFFOH = O.H. IN BITS FOR EF
EFEOH = O.H. IN BITS FOR EACH LE IN EF
EFNOC = NO. OF OCCURENCES OF EF FOR SPECIFIED RFO AND RFQ
EFRFO = RELATIONAL/FILE OCCURENCE (RFO) INDICATOR
EFRFQ = RELATIONAL/FILE QUALIFIER (RFQ)
EFBUF = I/O SOFTWARE BUFFER SIZE IN BIOUS

INDEX*	EFNAM	EFLET	EFIXZ	EFFOH	EFEOH	EFNOC	EFRFO	EFRFQ	EFBUF
1	DMAIN	2	Ø	Ø	Ø	1			5
2	ALIAS	2	Ø	18Ø	Ø	1			5
3	ELMNT	5	Ø	18Ø	Ø	1			5
4	ETNDX	1	11Ø88Ø	216	Ø	1			5
5	TXTBL	2	Ø	18Ø	Ø	1			5
6	SNDXM	1	36864	Ø	Ø	1			2
7	SWORK	2	Ø	36	*FXLB	1			2
8	SINVS	1	Ø	36	*FXLB	1	RLX		2
9	SRAVI	2	Ø	*AVOH	Ø	1	ATX		2
1Ø	TEMEF	1	Ø	Ø	Ø	1			5

PARAMETER FOR EF LOGICAL ENTRIES

LENAM = NAME OF LOGICAL ENTRY (LE)
LEFUN = FUNCTIONAL TYPE FOR LE
LEFRF = ELEMENTARY FILE REFERENCE
LENOC = NO. OF OCCURENCES OF LE FOR SPECIFIED RFO AND RFQ
LERFO = RELATIONAL/FILE OCCURENCE INDICATOR (RFO)
LERFQ = RELATIONAL/FILE QUALIFIER (RFQ)
LESIZ = SIZE IN BITS OF LE
LEOHD = O.H. IN BITS FOR LE OCCURENCE

INDEX*	LENAM	LEFUN	LEFRF	LENOC	LERFO	LERFQ	LESIZ	LEOHD
1	ATDEF	DEF	DMAIN	1	ATR		576	Ø
2	REDEF	DEF	DMAIN	1	RL		576	Ø
3	ATALI	DEF	ALIAS	2	ATD		72Ø	Ø
4	RLALI	DEF	ALIAS	1	RL		72Ø	Ø
5	SETPI	PRI	ELMNT	1			72Ø	Ø
6	ATPID	PRI	ELMNT	1	ATD		72Ø	Ø
7	ATOM	INS	ELMNT	1	AID		72Ø	Ø
8	TUPLE	PRI	ELMNT	1	TU		72Ø	Ø
9	RELAT	PRI	ELMNT	1	RL		72Ø	Ø
1Ø	EHASH	SEC	ETNDX	1	LE	ELMNT	36	Ø
11	SPNTR	SEC	SNDXM	1	RLX		46Ø8	36
12	UNVRS	SEC	SWORK	1	RLX		*IRBS	54Ø
13	WRKLE	TMP	SWORK	1			Ø	Ø
14	AVSET	SEC	SINVS	1	AIXA		*IIBS	54Ø
15	RAVAL	SEC	SRAVI	1	AIXA		36	Ø
16	RAPTR	SEC	SRAVI	1	AIXA		36	Ø
17	RLBST	PRI	TXTBL	1	RL		*PBST	72Ø
18	TUBST	PRI	TXTBL	1	TU		*PBST	72Ø
19	TEMLE	OVH	TEMEF	1			Ø	Ø

HARDWARE/SOFTWARE ENVIRONMENT PARAMETERS

ENPPI = PROCESSOR POWER INDICATOR: 1 = NBS ECF PDP/1Ø
ENSAU = SIZE IN BITS OF SMALLEST ADDRESSABLE UNIT (SAU)
ENCPS = NUMBER OF CHARACTERS PER SAU
ENBIU = SIZE IN SAU'S OF BASIC I/O UNIT (BIOU)
ENIOM = MAX NO. OF BIOU'S THAT CAN BE TRANSFERRED WITH A SINGLE ACCESS
ENIRA = ACCESS TIME IN MILLISECS FOR SECONDARY STORAGE READ
ENIWA = ACCESS TIME IN MILLISECS FOR SECONDARY STORAGE WRITE
ENIRT = TRANSFER TIME IN MILLISECS FOR SECONDARY STORAGE READ
ENIWT = TRANSFER TIME IN MILLISECS FOR SECONDARY STORAGE WRITE
ENIOP = TIME IN MILLISECS TO OPEN SECONDARY STORAGE FILE
ENICL = TIME IN MILLISECS TO CLOSE SECONDARY STORAGE FILE
ENIDE = TIME IN MILLISECS TO DELETE SECONDARY STORAGE FILE

ENPPI	ENLOD	ENSAU	ENCPS	ENBIU	ENIOM	ENIRA	ENIWA	ENIRT	ENIWT	ENIOP	ENICL	ENIDE
1.ØØ	1.ØØ	36	5	128	5	23	9Ø	51	1Ø4	25	25	371Ø

GLOBAL SOFTWARE PARAMETERS

GSNFN = NO. OF DBMS FUNCTIONS DEFINED

GSNFN
33

SOFTWARE FUNCTION PARAMETERS

FNNAM = NAME OF DBMS FUNCTION
FNPRC = MILLISECS OF PROCESS TIME FOR EACH EXECUTION OF FUNCTION
FNMOD = MODIFICATION FACTOR FOR DBMS FUNCTION PROCESSOR TIME

INDEX	FNNAM	FNPRC	FNMOD
1	M9COM	236	Ø.ØØ
2	S9PRO	834	Ø.ØØ
3	M9PID	12	Ø.ØØ
4	M9ISO	3	Ø.ØØ
5	M9GTL	4	Ø.ØØ
6	M9GTS	1Ø	Ø.ØØ
7	M9GTR	5	Ø.ØØ
8	A7TRA	8	Ø.ØØ
9	A7SRC	134	Ø.ØØ
1Ø	ALLOC	8	Ø.ØØ
11	DALOC	5	Ø.ØØ
12	SSAVE	8	Ø.ØØ
13	SDEST	5	Ø.ØØ
14	UNION	Ø	Ø.ØØ
15	INTRS	Ø	Ø.ØØ
16	XUNSD	Ø	Ø.ØØ
17	RLCMP	Ø	Ø.ØØ
18	SCOPY	Ø	Ø.ØØ
19	RGSTR	41	Ø.ØØ
2Ø	SADD1	116	Ø.ØØ
21	S9PTR	28	Ø.ØØ
22	M9ALO	9	Ø.ØØ
23	S4CHK	12	Ø.ØØ
24	S4SUB	53	Ø.ØØ
25	S4EVA	3Ø	Ø.ØØ
26	S4MOV	13	Ø.ØØ
27	S4SRC	3	Ø.ØØ
28	S4IDX	8Ø	Ø.ØØ
29	S4BLD	5	Ø.ØØ
3Ø	S4OPR	5	Ø.ØØ
31	GPTPK	8	Ø.ØØ
32	TRVRS	7	Ø.ØØ
33	K9IO	12	Ø.ØØ

BIT STRING PROCESSING PARAMETERS

BSPKS = NO. OF BITS IN QUATREE PACKET
BSQLV = NO. OF QUATREE LEVELS
BSMPI = MAX NO. OF POSITION ID'S FOR FILE STRUCTURE
BSYNT = Y INTERCEPT FOR TRAVERSAL ESTIMATION
BSX1C = X1 COEF FOR TRAVERSAL SET CARDINALITY
BSX2C = X2 COEF FOR TRAVERSAL SET RANGE

BSPKS	BSQLV	BSMPI	BSYNT	BSX1C	BSX2C
4	16	9			

M>
R

SPPM - RESPONSE TIME ESTIMATION MODELER

GROSS PARAMETERS FOR MODELING QUERIES ON DB VGH5 :
* 1 DEFINED RELATIONS
* 10 DOMAIN DEFINITIONS
* 10 ATTRIBUTE DEFINITIONS
* 10 ELEMENTARY FILES DEFINED
* 19 LOGICAL ENTRY TYPES DEFINED
* 33 DBMS FUNCTIONS DEFINED
* 1.00 SYSTEM LOAD FACTOR
* 1.00 PROCESSOR POWER INDICATOR

ENTER FILE NAME FOR RESPONSE TIME ESTIMATION REPORT
VB5NEW
OUTPUT WILL APPEAR IN FILE - VB5NEW.RES

```
READY TO ACCEPT SPP QUERIES FOR VGH5  RELATIONS:

    RELATION NAME  NO ATTS  NO TUPLES
    ---------------------------------
        REL5          1Ø        7

USE SPP QUERY FORMAT: EXIT RETURNS TO MODEL DRIVER


C>
SUBX *A=REL5(X.(*):X.NAME.EQ.TWA.AND.X.IAS.GT.222.22);

S9PTRB MODELING SUBX PREDICATE EVALUATION:
    ENTER INTEGER NUMBER OF          7 TUPLES IN RELATION REL5
    THAT SATISFY EACH ELEMENTARY CONDITION, OR
    ENTER  "?"  FOR MODEL DETERMINATION
**  1-ST COND: NAME   .EQ. TWA  ...

RGSTR MODELING REGISTRATION OF ATOM:
**IS STRING TWA  ...IN DB?  ENTER (Y OR N).

:>
Y
ENTER NO IN RANGE     Ø TO      2 , OR ENTER "?" >
2
**  2-ND COND: IAS   .GT.  222.2

RGSTR MODELING REGISTRATION OF ATOM:
**IS VALUE  222.2    IN DB?  ENTER (Y OR N).

:>
Y
ENTER NO IN RANGE     Ø TO      7 , OR ENTER "?" >
5
```

RESPONSE SUMMARY

DESC	QUERY		SESSION	
	MS	%	MS	%
I/O	981.ØØ	17.39	981.ØØ	17.39
PROCESSING	4659.Ø	82.61	4659.Ø	82.61
OVERHEAD	Ø.ØØØØØE+ØØ	Ø.ØØ	Ø.ØØØØØE+ØØ	Ø.ØØ
RESPONSE	564Ø.Ø	1ØØ.ØØ	564Ø.Ø	1ØØ.ØØ

I/O SUMMARY

DESC	QUERY	SESSION
NO PHYSICAL READS	5	5
NO PHYSICAL WRITES	Ø	Ø
NO OTHER I/O'S		
OPEN	Ø	Ø
CLOSE	Ø	Ø
DELETE	Ø	Ø
NO BIOU'S TRANS	16	16
ACCESS TIME	115.ØØ / 11.72%	115.ØØ / 11.72%
TRANSFER TIME	816.ØØ / 83.18%	816.ØØ / 83.18%
OTHER I/O TIME	5Ø.ØØ / 5.1Ø%	5Ø.ØØ / 5.1Ø%
TOTAL TIME	981.ØØ	981.ØØ

```
C>
SUBX *B=REL5(X.(*):X.NAME.EQ.CLYDE.OR.X.IAS.LT.12.Ø.AND.X.Q.GT.23);

S9PTRB MODELING SUBX PREDICATE EVALUATION:
    ENTER INTEGER NUMBER OF          7 TUPLES IN RELATION REL5
    THAT SATISFY EACH ELEMENTARY CONDITION, OR
    ENTER  "?"  FOR MODEL DETERMINATION
**  1-ST COND: NAME  .EQ. CLYDE...

RGSTR MODELING REGISTRATION OF ATOM:
**IS STRING CLYDE...IN DB?  ENTER (Y OR N).

:>
Y
ENTER NO IN RANGE     Ø TO     2 , OR ENTER "?" >
1
**  2-ND COND: IAS   .LT.  12.ØØ

RGSTR MODELING REGISTRATION OF ATOM:
**IS VALUE  12.ØØ    IN DB?  ENTER (Y OR N).

:>
Y
ENTER NO IN RANGE     Ø TO     7 , OR ENTER "?" >
4
**  3-RD COND: Q     .GT.  23.ØØ

RGSTR MODELING REGISTRATION OF ATOM:
**IS VALUE  23.ØØ    IN DB?  ENTER (Y OR N).

:>
Y
ENTER NO IN RANGE     Ø TO     7 , OR ENTER "?" >
7
```

RESPONSE SUMMARY

DESC	QUERY		SESSION	
	MS	%	MS	%
I/O	556.00	6.15	1537.0	10.47
PROCESSING	8478.0	93.85	13137.	89.53
OVERHEAD	0.00000E+00	0.00	0.00000E+00	0.00
RESPONSE	9034.0	100.00	14674.	100.00

I/O SUMMARY

DESC	QUERY		SESSION	
NO PHYSICAL READS	2		7	
NO PHYSICAL WRITES	0		0	
NO OTHER I/O'S				
OPEN	0		0	
CLOSE	0		0	
DELETE	0		0	
NO BIOU'S TRANS	10		26	
ACCESS TIME	46.00 /	8.27%	161.00 /	10.47%
TRANSFER TIME	510.00 /	91.73%	1326.00 /	86.27%
OTHER I/O TIME	0.00 /	0.00%	50.00 /	3.25%
TOTAL TIME	556.00		1537.00	

```
C>
SUBX *C=REL5(X.(*):X.IAS.LT.22.22.OR.X.Q.GT.2Ø.Ø.OR.X.NAME.EQ.DON)

S9PTRB MODELING SUBX PREDICATE EVALUATION:
    ENTER INTEGER NUMBER OF          7 TUPLES IN RELATION REL5
    THAT SATISFY EACH ELEMENTARY CONDITION, OR
    ENTER  "?"  FOR MODEL DETERMINATION
**  1-ST COND: IAS   .LT.  22.22

RGSTR MODELING REGISTRATION OF ATOM:
**IS VALUE  22.22    IN DB?  ENTER (Y OR N).

:>
Y
ENTER NO IN RANGE     Ø TO     7 , OR ENTER "?" >
3
**  2-ND COND: Q     .GT.  2Ø.ØØ

RGSTR MODELING REGISTRATION OF ATOM:
**IS VALUE  2Ø.ØØ    IN DB?  ENTER (Y OR N).

:>
Y
ENTER NO IN RANGE     Ø TO     7 , OR ENTER "?" >
5
**  3-RD COND: NAME  .EQ. DON  ...

RGSTR MODELING REGISTRATION OF ATOM:
**IS STRING DON  ...IN DB?  ENTER (Y OR N).

:>
Y
ENTER NO IN RANGE     Ø TO     2 , OR ENTER "?" >
1
```

RESPONSE SUMMARY

DESC	QUERY MS	QUERY %	SESSION MS	SESSION %
I/O	556.ØØ	6.56	2Ø93.Ø	9.Ø4
PROCESSING	7918.Ø	93.44	21Ø55.	9Ø.96
OVERHEAD	Ø.ØØØØØE+ØØ	Ø.ØØ	Ø.ØØØØØE+ØØ	Ø.ØØ
RESPONSE	8474.Ø	1ØØ.ØØ	23148.	1ØØ.ØØ

I/O SUMMARY

DESC	QUERY	SESSION
NO PHYSICAL READS	2	9
NO PHYSICAL WRITES	Ø	Ø
NO OTHER I/O'S		
OPEN	Ø	Ø
CLOSE	Ø	Ø
DELETE	Ø	Ø
NO BIOU'S TRANS	1Ø	36
ACCESS TIME	46.ØØ / 8.27%	2Ø7.ØØ / 9.89%
TRANSFER TIME	51Ø.ØØ / 91.73%	1836.ØØ / 87.72%
OTHER I/O TIME	Ø.ØØ / Ø.ØØ%	5Ø.ØØ / 2.39%
TOTAL TIME	556.ØØ	2Ø93.ØØ

```
C>
EXIT;

M>
X
STOP

END OF EXECUTION
CPU TIME: 58.53 ELAPSED TIME: 2:6.98
 EXIT
```

.TYPE ZLFNEW.SIZ

SPPM - POSITIONAL SET PROCESSOR PERFORMANCE MODEL

SIZE ESTIMATION ANALYSES

ESTIMATES FOR DB ZLOOF GENERATED ON 14-Feb-79 AT 1Ø:57

SOURCE DATABASE SIZE ANALYSIS

RELNAM	ALL DEFINED RELATIONS: ATTRIBUTES NAMED	ATTRIBUTES TOTAL	INDXD	TUPLES	REL SIZE (BITS)	TOTAL DATABASE: REL REPL	ATTRIBUTES TOTAL	INDXED	TUPLES	DATABASE TOTAL SIZE	TOTAL BIOUS
EMP	4	4	2	1Ø.ØØØ	828Ø.Ø	1	4	2	1Ø.ØØØ	828Ø.Ø	2
SALES	2	2	Ø	12.ØØØ	1296.Ø	1	2	Ø	12.ØØØ	1296.Ø	1
SUPLY	2	2	2	1Ø.ØØØ	756Ø.Ø	1	2	2	1Ø.ØØØ	756Ø.Ø	2
TYPE	3	3	1	9.ØØØØ	1296.Ø	1	3	1	9.ØØØØ	1296.Ø	1
TOTAL	11	11	5	41.ØØØ	18432.	4	11	5	41.ØØØ	18432.	6

STORED DATABASE SIZE - ELEMENTARY FILE ANALYSIS

E-FILE NO	NAME	NO LE TYPES	NUMBER LE'S	TOTAL EF SIZE	EF REPL	TOTAL ALL EF	STORED DATABASE SIZE BY STORAGE FUNCTION: PRIM REL	SECN REL	DEFN	INST DATA	OVERHEAD	TOTAL BIOUS
1	DMAIN	2	15.ØØØ	864Ø.Ø	1	864Ø.Ø			864Ø.			2
2	ALIAS	2	2Ø.ØØØ	1458Ø.	1	1458Ø.			.144ØE+Ø5		18Ø.Ø	4
3	ELMNT	5	94.ØØØ	6786Ø.	1	6786Ø.	.3888E+Ø5			.288ØE+Ø5	18Ø.Ø	15
4	ETNDX	1	94.ØØØ	.11Ø88E+Ø6	1	.11Ø88E+Ø6		3384.			.1Ø75E+Ø6	25
5	TXTBL	2	45.ØØØ	37187.	1	37187.	46Ø7.				.3258E+Ø5	9
6	SNDXM	1	3.ØØØØ	36864.	1	36864.		.1382E+Ø5			.23Ø4E+Ø5	8
7	SWORK	2	4.ØØØØ	1386Ø.	1	1386Ø.		354.Ø			.1351E+Ø5	4
8	SINVS	1	31.ØØØ	47652.	3	.14296E+Ø6		2232.			.14Ø7E+Ø6	33
9	SRAVI	2	62.ØØØ	959Ø.4	5	47952.		2232.			.4572E+Ø5	15
1Ø	TEMEF	1	1.ØØØØ	.ØØØØØE+ØØ	1	.ØØØØØE+ØØ						Ø
	TOTALS	19	369.ØØ	.34711E+Ø6	16	.48Ø78E+Ø6	43487.	22Ø26.	23Ø4Ø.	288ØØ.	.36343E+Ø6	115

STORED DATABASE SIZE - LOGICAL ENTRY ANALYSIS

NO	NAME	EF REF	SIZE(BITS)	STOFNC	LE OVHD	BITS/LE	LE OCCUR	LE FUNC	LE OVHD	TOTAL LE
1	ATDEF	DMAIN	576.ØØ	DEF		576.ØØ	11.ØØØ	6336.Ø		6336.Ø
2	REDEF	DMAIN	576.ØØ	DEF		576.ØØ	4.ØØØØ	23Ø4.Ø		23Ø4.Ø
3	ATALI	ALIAS	72Ø.ØØ	DEF		72Ø.ØØ	16.ØØØ	1152Ø.		1152Ø.
4	RLALI	ALIAS	72Ø.ØØ	DEF		72Ø.ØØ	4.ØØØØ	288Ø.Ø		288Ø.Ø
5	SETPI	ELMNT	72Ø.ØØ	PRI		72Ø.ØØ	1.ØØØØ	72Ø.ØØ		72Ø.ØØ
6	ATPID	ELMNT	72Ø.ØØ	PRI		72Ø.ØØ	8.ØØØØ	576Ø.Ø		576Ø.Ø
7	ATOM	ELMNT	72Ø.ØØ	INS		72Ø.ØØ	4Ø.ØØØ	288ØØ.		288ØØ.
8	TUPLE	ELMNT	72Ø.ØØ	PRI		72Ø.ØØ	41.ØØØ	2952Ø.		2952Ø.
9	RELAT	ELMNT	72Ø.ØØ	PRI		72Ø.ØØ	4.ØØØØ	288Ø.Ø		288Ø.Ø
1Ø	EHASH	ETNDX	36.ØØØ	SEC		36.ØØØ	94.ØØØ	3384.Ø		3384.Ø
11	SPNTR	SNDXM	46Ø8.Ø	SEC	36.ØØ	4644.Ø	3.ØØØØ	13824.	1Ø8.Ø	13932.
12	UNVRS	SWORK	118.ØØ	SEC	449Ø.	46Ø8.Ø	3.ØØØØ	354.ØØ	.1347E+Ø5	13824.
13	WRKLE	SWORK	.ØØØØØE+ØØ	TMP		.ØØØØØE+ØØ	1.ØØØØ	.ØØØØØE+ØØ		.ØØØØØE+ØØ
14	AVSET	SINVS	72.ØØØ	SEC	4536.	46Ø8.Ø	31.ØØØ	2232.Ø	.14Ø6E+Ø6	.14285E+Ø6
15	RAVAL	SRAVI	36.ØØØ	SEC		36.ØØØ	31.ØØØ	1116.Ø		1116.Ø
16	RAPTR	SRAVI	36.ØØØ	SEC		36.ØØØ	31.ØØØ	1116.Ø		1116.Ø
17	RLBST	TXTBL	219.ØØ	PRI	72Ø.Ø	939.ØØ	4.ØØØØ	876.ØØ	288Ø.	3756.Ø
18	TUBST	TXTBL	91.ØØØ	PRI	72Ø.Ø	811.ØØ	41.ØØØ	3731.Ø	.2952E+Ø5	33251.
19	TEMLE	TEMEF	.ØØØØØE+ØØ	OVH		.ØØØØØE+ØØ	1.ØØØØ	.ØØØØØE+ØØ		.ØØØØØE+ØØ
	TOTALS						369.ØØ	.11735E+Ø6	.18659E+Ø6	.3Ø395E+Ø6

.TYPE ZLFNEW.RES

SPPM - POSITIONAL SET PROCESSOR PERFORMANCE MODEL

RESPONSE TIME ESTIMATION

ESTIMATES FOR DB ZLOOF GENERATED ON 14-Feb-79 AT 10:57

SUBX *A=EMP(X.(*):X.NAME.EQ.JONES);

RESPONSE SUMMARY

DESC	QUERY		SESSION	
	MS	%	MS	%
I/O	981.00	56.35	981.00	56.35
PROCESSING	760.00	43.65	760.00	43.65
OVERHEAD	0.00000E+00	0.00	0.00000E+00	0.00
RESPONSE	1741.0	100.00	1741.0	100.00

I/O SUMMARY

DESC	QUERY	SESSION
NO PHYSICAL READS	5	5
NO PHYSICAL WRITES	0	0
NO OTHER I/O'S		
OPEN	0	0
CLOSE	0	0
DELETE	0	0
NO BIOU'S TRANS	16	16
ACCESS TIME	115.00 / 11.72%	115.00 / 11.72%
TRANSFER TIME	816.00 / 83.18%	816.00 / 83.18%
OTHER I/O TIME	50.00 / 5.10%	50.00 / 5.10%
TOTAL TIME	981.00	981.00

```
SUBX *B=SUPLY(X.(*):X.ITEM.GT.200);
```

RESPONSE SUMMARY

DESC	QUERY		SESSION	
	MS	%	MS	%
I/O	0.00000E+00	0.00	981.00	6.45
PROCESSING	13480.	100.00	14240.	93.55
OVERHEAD	0.00000E+00	0.00	0.00000E+00	0.00
RESPONSE	13480.	100.00	15221.	100.00

I/O SUMMARY

DESC	QUERY	SESSION
NO PHYSICAL READS	0	5
NO PHYSICAL WRITES	0	0
NO OTHER I/O'S		
OPEN	0	0
CLOSE	0	0
DELETE	0	0
NO BIOU'S TRANS	0	16
ACCESS TIME	0.00 / 0.00%	115.00 / 11.72%
TRANSFER TIME	0.00 / 0.00%	816.00 / 83.18%
OTHER I/O TIME	0.00 / 0.00%	50.00 / 5.10%
TOTAL TIME	0.00	981.00

```
EXIT;
```

SESSION RESPONSE ANALYSIS BY DATABASE FUNCTION

DATABASE FUNCTION	NO EXEC	TIME IN MS PROCESS	TIME IN MS I/O	TOTAL TIME MS	TOTAL TIME %
SADD1	9	11050.00	0.00	11050.00	57.88
SSAVE	4	40.00	1692.00	1732.00	11.38
SCOPY	3	1140.00	0.00	1140.00	7.49
RLCMP	1	820.00	0.00	820.00	5.39
ALLOC	10	100.00	590.00	690.00	4.53
RGSTR	4	40.00	576.00	616.00	4.05
UNION	1	570.00	0.00	570.00	3.74
S4MOV	4	40.00	320.00	360.00	2.37
M9ISO	9	90.00	0.00	90.00	0.59
A7SRC	4	40.00	0.00	40.00	0.26
A7TRA	4	40.00	0.00	40.00	0.26
M9COM	3	30.00	0.00	30.00	0.20
S4IDX	2	20.00	0.00	20.00	0.13
S4SRC	2	20.00	0.00	20.00	0.13
S4EVA	2	20.00	0.00	20.00	0.13
S4SUB	2	20.00	0.00	20.00	0.13
S4CHK	2	20.00	0.00	20.00	0.13
M9ALO	2	20.00	0.00	20.00	0.13
S9PTR	2	20.00	0.00	20.00	0.13
SDEST	2	20.00	0.00	20.00	0.13
DALOC	2	20.00	0.00	20.00	0.13
M9GTS	2	20.00	0.00	20.00	0.13
M9GTL	2	20.00	0.00	20.00	0.13
M9PID	2	20.00	0.00	20.00	0.13
S9PRO	2	20.00	0.00	20.00	0.13
S4BLD	1	10.00	0.00	10.00	0.07
(OTHER)	0	0.00	0.00	0.00	0.00
TOTAL	83	14270.00	3178.00	17448.00	100.00
OVERHEAD				0.00	0.00
TOTAL RESPONSE				17448.00	100.00

SESSION I/O ANALYSIS BY ELEMENTARY FILE

E-FILE	NO. PHYSICAL I/O REQUESTS			NO BIOU'S	TOTAL I/O TIME	
	READ	WRITE	OTHER	TRAN	MS	%
ELMNT	2	Ø	Ø	1Ø	556.ØØ	56.68
SRAVI	1	Ø	Ø	2	15Ø.ØØ	15.29
SINVS	1	Ø	Ø	2	15Ø.ØØ	15.29
SWORK	1	Ø	Ø	2	125.ØØ	12.74
TOTAL	5	Ø	Ø	16	981.ØØ	1ØØ.ØØ

.TYPE VB5NEW.SIZ

SPPM - POSITIONAL SET PROCESSOR PERFORMANCE MODEL

SIZE ESTIMATION ANALYSES

ESTIMATES FOR DB VGH5 GENERATED ON 14-Feb-79 AT Ø9:49

SOURCE DATABASE SIZE ANALYSIS

RELNAM	ALL DEFINED RELATIONS					TOTAL DATABASE					
	ATTRIBUTES NAMED	ATTRIBUTES TOTAL	INDXD	TUPLES	REL SIZE (BITS)	REL REPL	ATTRIBUTES TOTAL	ATTRIBUTES INDXED	TUPLES	DATABASE TOTAL SIZE	TOTAL BIOUS
REL5	1Ø	1Ø	5	7.ØØØØ	2772.Ø	1	1Ø	5	7.ØØØØ	2772.Ø	1
TOTAL	1Ø	1Ø	5	7.ØØØØ	2772.Ø	1	1Ø	5	7.ØØØØ	2772.Ø	1

STORED DATABASE SIZE - ELEMENTARY FILE ANALYSIS

E-FILE NO	NAME	NO LE TYPES	NUMBER LE'S	TOTAL EF SIZE	EF REPL	TOTAL ALL EF	.........STORED DATABASE SIZE BY STORAGE FUNCTION......... PRIM REL	SECN REL	DEFN	INST DATA	OVERHEAD	TOTAL BIOUS
1	DMAIN	2	11.000	6336.0	1	6336.0			6336.			2
2	ALIAS	2	21.000	15300.	1	15300.			.1512E+05		180.0	4
3	ELMNT	5	51.000	36900.	1	36900.	.1368E+05			.2304E+05	180.0	9
4	ETNDX	1	51.000	.11088E+06	1	.11088E+06		1836.			.1090E+06	25
5	TXTBL	2	8.0000	7293.0	1	7293.0	1353.				5940.	2
6	SNDXM	1	1.0000	36864.	1	36864.		4608.			.3226E+05	8
7	SWORK	2	2.0000	4644.0	1	4644.0		113.0			4531.	2
8	SINVS	1	21.000	96804.	1	96804.		2499.			.9431E+05	22
9	SRAVI	2	42.000	9590.4	5	47952.		1512.			.4644E+05	15
10	TEMEF	1	1.0000	.00000E+00	1	.00000E+00						0
	TOTALS	19	209.00	.32461E+06	14	.36297E+06	15033.	10568.	21456.	23040.	.29288E+06	89

STORED DATABASE SIZE - LOGICAL ENTRY ANALYSIS

NO	NAME	EF REF	SIZE(BITS)	STOFNC	LE OVHD	BITS/LE	LE OCCUR	LE FUNC	LE OVHD	TOTAL LE
1	ATDEF	DMAIN	576.00	DEF		576.00	10.000	5760.0		5760.0
2	REDEF	DMAIN	576.00	DEF		576.00	1.0000	576.00		576.00
3	ATALI	ALIAS	720.00	DEF		720.00	20.000	14400.		14400.
4	RLALI	ALIAS	720.00	DEF		720.00	1.0000	720.00		720.00
5	SETPI	ELMNT	720.00	PRI		720.00	1.0000	720.00		720.00
6	ATPID	ELMNT	720.00	PRI		720.00	10.000	7200.0		7200.0
7	ATOM	ELMNT	720.00	INS		720.00	32.000	23040.		23040.
8	TUPLE	ELMNT	720.00	PRI		720.00	7.0000	5040.0		5040.0
9	RELAT	ELMNT	720.00	PRI		720.00	1.0000	720.00		720.00
10	EHASH	ETNDX	36.000	SEC		36.000	51.000	1836.0		1836.0
11	SPNTR	SNDXM	4608.0	SEC	36.00	4644.0	1.0000	4608.0	36.00	4644.0
12	UNVRS	SWORK	113.00	SEC	4495.	4608.0	1.0000	113.00	4495.	4608.0
13	WRKLE	SWORK	.00000E+00	TMP		.00000E+00	1.0000	.00000E+00		.00000E+00
14	AVSET	SINVS	119.00	SEC	4489.	4608.0	21.000	2499.0	.9427E+05	96768.
15	RAVAL	SRAVI	36.000	SEC		36.000	21.000	756.00		756.00
16	RAPTR	SRAVI	36.000	SEC		36.000	21.000	756.00		756.00
17	RLBST	TXTBL	184.00	PRI	720.0	904.00	1.0000	184.00	720.0	904.00
18	TUBST	TXTBL	167.00	PRI	720.0	887.00	7.0000	1169.0	5040.	6209.0
19	TEMLE	TEMEF	.00000E+00	OVH		.00000E+00	1.0000	.00000E+00		.00000E+00
	TOTALS						209.00	70097.	.10456E+06	.17466E+06

.TYPE VB5NEW.RES

SPPM - POSITIONAL SET PROCESSOR PERFORMANCE MODEL

RESPONSE TIME ESTIMATION

ESTIMATES FOR DB VGH5 GENERATED ON 14-Feb-79 AT 10:58

SUBX *A=REL5(X.(*):X.NAME.EQ.TWA.AND.X.IAS.GT.222.22);

RESPONSE SUMMARY

DESC	QUERY		SESSION	
	MS	%	MS	%
I/O	981.00	17.39	981.00	17.39
PROCESSING	4659.0	82.61	4659.0	82.61
OVERHEAD	0.00000E+00	0.00	0.00000E+00	0.00
RESPONSE	5640.0	100.00	5640.0	100.00

I/O SUMMARY

DESC	QUERY	SESSION
NO PHYSICAL READS	5	5
NO PHYSICAL WRITES	0	0
NO OTHER I/O'S		
OPEN	0	0
CLOSE	0	0
DELETE	0	0
NO BIOU'S TRANS	16	16
ACCESS TIME	115.00 / 11.72%	115.00 / 11.72%
TRANSFER TIME	816.00 / 83.18%	816.00 / 83.18%
OTHER I/O TIME	50.00 / 5.10%	50.00 / 5.10%
TOTAL TIME	981.00	981.00

```
SUBX *B=REL5(X.(*):X.NAME.EQ.CLYDE.OR.X.IAS.LT.12.Ø.AND.X.Q.GT.23);
```

RESPONSE SUMMARY

DESC	QUERY		SESSION	
	MS	%	MS	%
I/O	556.ØØ	6.15	1537.Ø	1Ø.47
PROCESSING	8478.Ø	93.85	13137.	89.53
OVERHEAD	Ø.ØØØØØE+ØØ	Ø.ØØ	Ø.ØØØØØE+ØØ	Ø.ØØ
RESPONSE	9Ø34.Ø	1ØØ.ØØ	14674.	1ØØ.ØØ

I/O SUMMARY

DESC	QUERY	SESSION
NO PHYSICAL READS	2	7
NO PHYSICAL WRITES	Ø	Ø
NO OTHER I/O'S		
OPEN	Ø	Ø
CLOSE	Ø	Ø
DELETE	Ø	Ø
NO BIOU'S TRANS	1Ø	26
ACCESS TIME	46.ØØ / 8.27%	161.ØØ / 1Ø.47%
TRANSFER TIME	51Ø.ØØ / 91.73%	1326.ØØ / 86.27%
OTHER I/O TIME	Ø.ØØ / Ø.ØØ%	5Ø.ØØ / 3.25%
TOTAL TIME	556.ØØ	1537.ØØ

SUBX *C=REL5(X.(*):X.IAS.LT.22.22.OR.X.Q.GT.2Ø.Ø.OR.X.NAME.EQ.DON);

RESPONSE SUMMARY

DESC	QUERY		SESSION	
	MS	%	MS	%
I/O	556.ØØ	6.56	2Ø93.Ø	9.Ø4
PROCESSING	7918.Ø	93.44	21Ø55.	9Ø.96
OVERHEAD	Ø.ØØØØØE+ØØ	Ø.ØØ	Ø.ØØØØØE+ØØ	Ø.ØØ
RESPONSE	8474.Ø	1ØØ.ØØ	23148.	1ØØ.ØØ

I/O SUMMARY

DESC	QUERY	SESSION
NO PHYSICAL READS	2	9
NO PHYSICAL WRITES	Ø	Ø
NO OTHER I/O'S		
OPEN	Ø	Ø
CLOSE	Ø	Ø
DELETE	Ø	Ø
NO BIOU'S TRANS	1Ø	36
ACCESS TIME	46.ØØ / 8.27%	2Ø7.ØØ / 9.89%
TRANSFER TIME	51Ø.ØØ / 91.73%	1836.ØØ / 87.72%
OTHER I/O TIME	Ø.ØØ / Ø.ØØ%	5Ø.ØØ / 2.39%
TOTAL TIME	556.ØØ	2Ø93.ØØ

EXIT;

SESSION RESPONSE ANALYSIS BY DATABASE FUNCTION

DATABASE FUNCTION	NO EXEC	TIME IN MS		TOTAL TIME	
		PROCESS	I/O	MS	%
SSAVE	23	184.00	8179.00	8363.00	31.13
SCOPY	20	5560.00	0.00	5560.00	15.02
UNION	10	4274.00	0.00	4274.00	11.46
RLCMP	5	2745.00	0.00	2745.00	6.86
S9PRO	3	2502.00	0.00	2502.00	6.81
RGSTR	11	451.00	1740.00	2141.00	5.47
A7SRC	11	1474.00	0.00	1474.00	4.37
ALLOC	44	352.00	760.00	1112.00	4.10
INTRS	2	1082.00	0.00	1082.00	4.07
M9COM	4	944.00	0.00	944.00	4.01
SADD1	4	810.00	0.00	810.00	3.50
S4IDX	8	640.00	0.00	640.00	2.76
S4MOV	16	208.00	162.00	370.00	1.60
S4SUB	3	159.00	0.00	159.00	0.69
S4EVA	3	90.00	0.00	90.00	0.39
A7TRA	11	88.00	0.00	88.00	0.38
S9PTR	3	84.00	0.00	84.00	0.36
SDEST	15	75.00	0.00	75.00	0.32
DALOC	10	50.00	0.00	50.00	0.22
S4CHK	3	36.00	0.00	36.00	0.16
M9PID	3	36.00	0.00	36.00	0.16
M9GTS	3	30.00	0.00	30.00	0.13
M9ALO	3	27.00	0.00	27.00	0.12
S4OPR	5	25.00	0.00	25.00	0.11
S4BLD	5	25.00	0.00	25.00	0.11
S4SRC	8	24.00	0.00	24.00	0.10
M9GTL	3	12.00	0.00	12.00	0.05
M9ISO	4	12.00	0.00	12.00	0.05
(OTHER)	0	0.00	0.00	0.00	0.00
TOTAL	243	21999.00	20841.00	32840.00	100.00
OVERHEAD				0.00	0.00
TOTAL RESPONSE				32840.00	100.00

SESSION I/O ANALYSIS BY ELEMENTARY FILE

-FILE	NO. PHYSICAL I/O REQUESTS			NO BIOU'S	TOTAL I/O TIME	
	READ	WRITE	OTHER	TRAN	MS	%
LMNT	6	Ø	Ø	3Ø	1668.ØØ	79.69
RAVI	1	Ø	Ø	2	15Ø.ØØ	7.17
INVS	1	Ø	Ø	2	15Ø.ØØ	7.17
WORK	1	Ø	Ø	2	125.ØØ	5.97
OTAL	9	Ø	Ø	36	2Ø93.ØØ	1ØØ.ØØ

☆ U.S. GOVERNMENT PRINTING OFFICE: 1979 O—281-067 (165)

NBS-114A (REV. 9-78)

U.S. DEPT. OF COMM. BIBLIOGRAPHIC DATA SHEET	1. PUBLICATION OR REPORT NO. SP 500-49	2. Gov't. Accession No.	3. Recipient's Accession No.

4. TITLE AND SUBTITLE	5. Publication Date
MODELING AND MEASUREMENT TECHNIQUES FOR EVALUATION OF DESIGN ALTERNATIVES IN THE IMPLEMENTATION OF DATABASE MANAGEMENT SOFTWARE	July 1979
	6. Performing Organization Code
7. AUTHOR(S) Donald R. Deutsch	8. Performing Organ. Report No.
9. PERFORMING ORGANIZATION NAME AND ADDRESS NATIONAL BUREAU OF STANDARDS DEPARTMENT OF COMMERCE WASHINGTON, DC 20234	10. Project/Task/Work Unit No. 6424200
	11. Contract/Grant No.
12. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS *(Street, City, State, ZIP)* Same as Item #9	13. Type of Report & Period Covered Final
	14. Sponsoring Agency Code

15. SUPPLEMENTARY NOTES

Library of Congress Catalog Card Number: 79-600088

☐ Document describes a computer program; SF-185, FIPS Software Summary, is attached.

16. ABSTRACT *(A 200-word or less factual summary of most significant information. If document includes a significant bibliography or literature survey, mention it here.)*

The substantial costs associated with building complex hardware/software systems make the traditional development approach of implementation followed by several iterations for modification and enhancement unacceptable for building modern database management systems. Mechanisms for determining gross feasibility prior to the commitment of resources for major software development efforts are required. An integrated approach combining the development of a limited but well-structured DBMS prototype with the use of high-level measurement and predictive modeling techniques for evaluating design alternatives in the implementation of database management software is proposed as an alternative to the traditional development-enhancement spiral.

Using a prototype for a set-theoretic implementation of a database management system with a relational user interface as an object, this research demonstrated that proposed DBMS designs can be evaluated through the use of performance prediction models based on prototype implementations and associated measurement systems.

17. KEY WORDS *(six to twelve entries; alphabetical order; capitalize only the first letter of the first key word unless a proper name; separated by semicolons)*

Analytic models; database management; model validation; performance evaluation; performance measurement; predictive modeling; set-processing; simulation; software design.

18. AVAILABILITY	19. SECURITY CLASS (THIS REPORT)	21. NO. OF PRINTED PAGES
XX Unlimited	UNCLASSIFIED	244
☐ For Official Distribution. Do Not Release to NTIS	20. SECURITY CLASS (THIS PAGE)	22. Price
XX Order From Sup. of Doc., U.S. Government Printing Office, Washington, DC 20402, SD Stock No. SN003-003-02088-5	UNCLASSIFIED	$5.50
☐ Order From National Technical Information Service (NTIS), Springfield, VA. 22161		

USCOMM-DC

ANNOUNCEMENT OF NEW PUBLICATIONS ON COMPUTER SCIENCE & TECHNOLOGY

Superintendent of Documents,
Government Printing Office,
Washington, D. C. 20402

Dear Sir:

Please add my name to the announcement list of new publications to be issued in the series: National Bureau of Standards Special Publication 500-.

Name ______________________________

Company ______________________________

Address ______________________________

City ______________ State ______________ Zip Code ______________

(Notification key N-503)

NBS TECHNICAL PUBLICATIONS

PERIODICALS

JOURNAL OF RESEARCH—The Journal of Research of the National Bureau of Standards reports NBS research and development in those disciplines of the physical and engineering sciences in which the Bureau is active. These include physics, chemistry, engineering, mathematics, and computer sciences. Papers cover a broad range of subjects, with major emphasis on measurement methodology, and the basic technology underlying standardization. Also included from time to time are survey articles on topics closely related to the Bureau's technical and scientific programs. As a special service to subscribers each issue contains complete citations to all recent NBS publications in NBS and non-NBS media. Issued six times a year. Annual subscription: domestic $17.00; foreign $21.25. Single copy, $3.00 domestic; $3.75 foreign.

Note: The Journal was formerly published in two sections: Section A "Physics and Chemistry" and Section B "Mathematical Sciences."

DIMENSIONS/NBS

This monthly magazine is published to inform scientists, engineers, businessmen, industry, teachers, students, and consumers of the latest advances in science and technology, with primary emphasis on the work at NBS. The magazine highlights and reviews such issues as energy research, fire protection, building technology, metric conversion, pollution abatement, health and safety, and consumer product performance. In addition, it reports the results of Bureau programs in measurement standards and techniques, properties of matter and materials, engineering standards and services, instrumentation, and automatic data processing.

Annual subscription: Domestic, $11.00; Foreign $13.75

NONPERIODICALS

Monographs—Major contributions to the technical literature on various subjects related to the Bureau's scientific and technical activities.

Handbooks—Recommended codes of engineering and industrial practice (including safety codes) developed in cooperation with interested industries, professional organizations, and regulatory bodies.

Special Publications—Include proceedings of conferences sponsored by NBS, NBS annual reports, and other special publications appropriate to this grouping such as wall charts, pocket cards, and bibliographies.

Applied Mathematics Series—Mathematical tables, manuals, and studies of special interest to physicists, engineers, chemists, biologists, mathematicians, computer programmers, and others engaged in scientific and technical work.

National Standard Reference Data Series—Provides quantitative data on the physical and chemical properties of materials, compiled from the world's literature and critically evaluated. Developed under a world-wide program coordinated by NBS. Program under authority of National Standard Data Act (Public Law 90-396).

NOTE: At present the principal publication outlet for these data is the Journal of Physical and Chemical Reference Data (JPCRD) published quarterly for NBS by the American Chemical Society (ACS) and the American Institute of Physics (AIP). Subscriptions, reprints, and supplements available from ACS, 1155 Sixteenth St. N.W., Wash., D.C. 20056.

Building Science Series—Disseminates technical information developed at the Bureau on building materials, components, systems, and whole structures. The series presents research results, test methods, and performance criteria related to the structural and environmental functions and the durability and safety characteristics of building elements and systems.

Technical Notes—Studies or reports which are complete in themselves but restrictive in their treatment of a subject. Analogous to monographs but not so comprehensive in scope or definitive in treatment of the subject area. Often serve as a vehicle for final reports of work performed at NBS under the sponsorship of other government agencies.

Voluntary Product Standards—Developed under procedures published by the Department of Commerce in Part 10, Title 15, of the Code of Federal Regulations. The purpose of the standards is to establish nationally recognized requirements for products, and to provide all concerned interests with a basis for common understanding of the characteristics of the products. NBS administers this program as a supplement to the activities of the private sector standardizing organizations.

Consumer Information Series—Practical information, based on NBS research and experience, covering areas of interest to the consumer. Easily understandable language and illustrations provide useful background knowledge for shopping in today's technological marketplace.

Order **above** *NBS publications from: Superintendent of Documents, Government Printing Office, Washington, D.C. 20402.*

Order **following** *NBS publications—NBSIR's and FIPS from the National Technical Information Services, Springfield, Va. 22161.*

Federal Information Processing Standards Publications (FIPS PUB)—Publications in this series collectively constitute the Federal Information Processing Standards Register. Register serves as the official source of information in the Federal Government regarding standards issued by NBS pursuant to the Federal Property and Administrative Services Act of 1949 as amended, Public Law 89-306 (79 Stat. 1127), and as implemented by Executive Order 11717 (38 FR 12315, dated May 11, 1973) and Part 6 of Title 15 CFR (Code of Federal Regulations).

NBS Interagency Reports (NBSIR)—A special series of interim or final reports on work performed by NBS for outside sponsors (both government and non-government). In general, initial distribution is handled by the sponsor; public distribution is by the National Technical Information Services (Springfield, Va. 22161) in paper copy or microfiche form.

BIBLIOGRAPHIC SUBSCRIPTION SERVICES

The following current-awareness and literature-survey bibliographies are issued periodically by the Bureau:

Cryogenic Data Center Current Awareness Service. A literature survey issued biweekly. Annual subscription: Domestic, $25.00; Foreign, $30.00.

Liquefied Natural Gas. A literature survey issued quarterly. Annual subscription: $20.00.

Superconducting Devices and Materials. A literature survey issued quarterly. Annual subscription: $30.00. Send subscription orders and remittances for the preceding bibliographic services to National Bureau of Standards, Cryogenic Data Center (275.02) Boulder, Colorado 80302.

U.S. DEPARTMENT OF COMMERCE
National Bureau of Standards
Washington, D.C. 20234

OFFICIAL BUSINESS

Penalty for Private Use, $300

POSTAGE AND FEES PAID
U.S. DEPARTMENT OF COMMERCE
COM-215

U.S.MAIL

SPECIAL FOURTH-CLASS RATE
BOOK