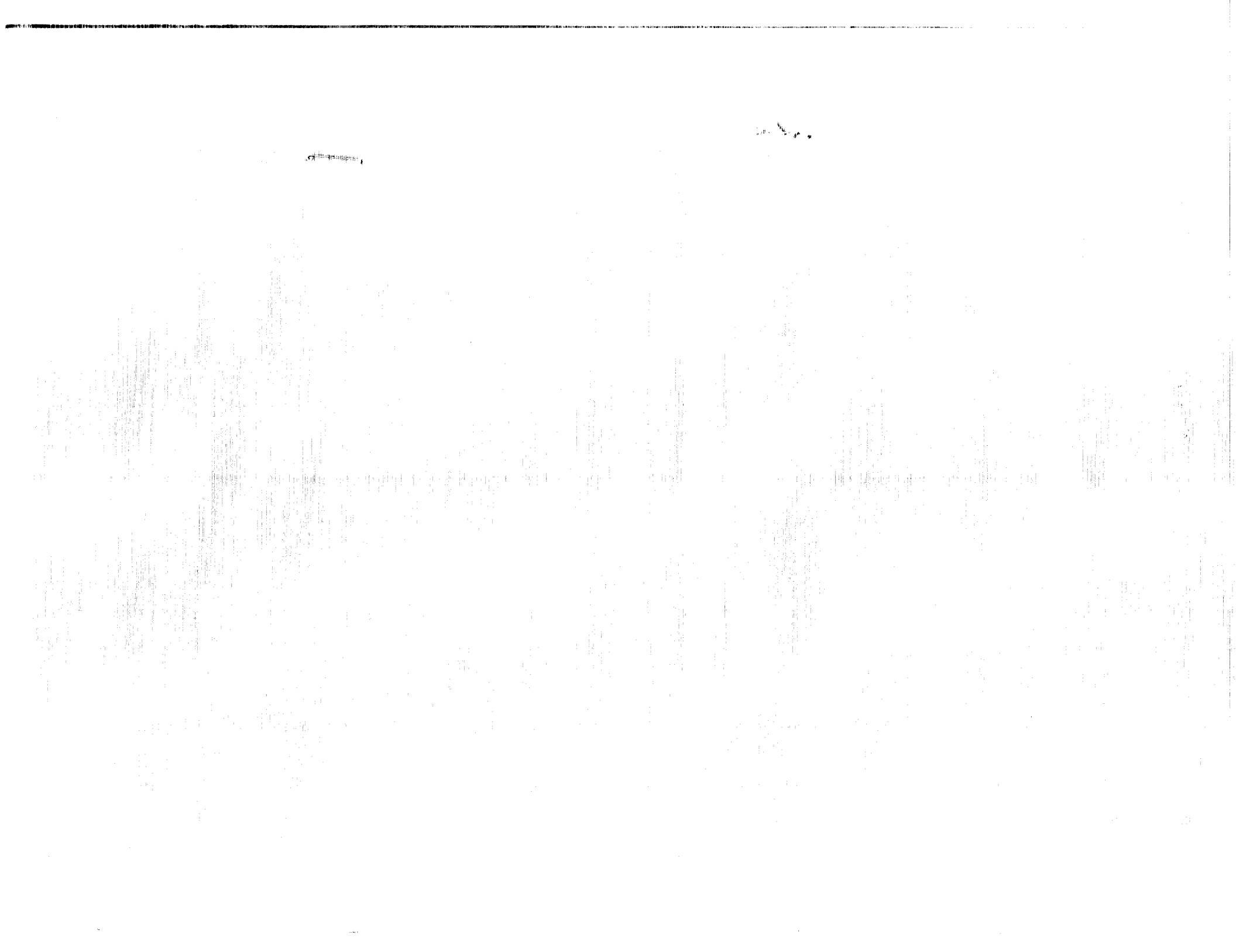
BORATORY SERIES

SEL-89-201

ARE ENGINEERING
ORATORY (SEL)
SE ORGANIZATION
USER'S GUIDE
REVISION 2

OCTOBER 1992

NASA



SOFTWARE ENGINEERING LABORATORY SERIES SEL-89-201

SOFTWARE ENGINEERING LABORATORY (SEL) DATABASE ORGANIZATION AND USER'S GUIDE REVISION 2

OCTOBER 1992

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

FOREWORD

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) and created to investigate the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1976 and has three primary organizational members:

NASA/GSFC, Software Engineering Branch

University of Maryland, Department of Computer Science

Computer Sciences Corporation, Software Engineering Operation

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effect of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document.

The original contributors to this document are

Maria So (Computer Sciences Corporation)
Gerard Heller (Computer Sciences Corporation)
Sandra Steinberg (Computer Sciences Corporation)
Karen Pumphrey (Computer Sciences Corporation)
Douglas Spiegel (NASA/GSFC)

The contributors to the latest revision of this document are

Linda Morusiewicz (Computer Sciences Corporation)
John Bristow (NASA/GSFC)

Single copies of this document can be obtained by writing to

Software Engineering Branch
Code 552
Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

This document presents the organization of the Software Engineering Laboratory (SEL) database. Included are definitions and detailed descriptions of the database tables and views, the SEL data, and system support data. The mapping from the SEL and system support data to the base tables is described. In addition, techniques for accessing the database through the Database Access Manager for the SEL (DAMSEL) system and via the ORACLE structured query language (SQL) are discussed.

TABLE OF CONTENTS

PAGE Vi INTENTIONALLY BLANK

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

SECTION 1—INTRODUCTION

The Software Engineering Laboratory (SEL) was established in 1976 to support research in measurement and evaluation of the software development process. Under its sponsorship, numerous experiments have been designed and executed to study the effects of applying various tools, methodologies, and models to software development efforts in flight dynamics applications. The SEL is a cooperative effort of the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC), Computer Sciences Corporation (CSC), and the University of Maryland.

To support the research activities it sponsors, one of the major functions of the SEL is the collection of detailed software engineering data, describing all facets of the development process, and the archival of this data for future use. To this end, the SEL has created and maintained an online database for the storage and retrieval of software engineering data. The SEL database has been designed and implemented as a relational database under the ORACLE relational database management system (RDBMS) on the Systems Technology Laboratory (STL) VAX 11/780 at GSFC. Since ORACLE provides the facilities for organizing, storing, maintaining, and retrieving data, SEL database users do not have to understand the physical organization of the data. They need only understand the logical structure of the database in order to query, calculate, and manipulate a variety of information. SEL database users include those involved in software engineering research, managers of current flight dynamics development efforts, and those involved in the collection of SEL data and maintenance of the database.

This document is intended as a reference guide for all SEL database users. Its purpose is to provide general users with high-level information about data collected by the SEL and how they are stored in the database. Information on how to access the data via various access paths is also provided. For database maintenance personnel, this document provides in-depth information about the structure of the database, including table and field definitions, indexes used, and constraints among data items.

Since this document is intended to be referenced by a broad spectrum of users, it is organized in increasing levels of specification. Section 1.1 describes general relational database concepts and terminology for readers who are not familiar with relational database systems. Section 2 of the document presents an introduction to the types of data that are stored from a conceptual point of view (i.e., without regard to physical or logical storage characteristics). Section 3 discusses the organization of the data with respect to their sources and the form in which they are collected. The conceptual view in Section 2 and the data collection view in Section 3 are then mapped into a logical view of the database design. This design is presented in Section 4. The logical design of the database is the lowest level of detail required to understand how to access the database. Details of the physical implementation are hidden from the user via the ORACLE RDBMS. Section 5 discusses various ways to actually access the SEL database. Appendix A lists all codes used in the database; Appendix B presents

sample database queries; Appendix C presents the SEL data collection forms; and Appendix D contains the data definition language (DDL), which specifies the definitions and constraints of the database tables and views.

1.1 BASIC RELATIONAL DATABASE CONCEPTS

In relational database terminology, the basic structure for storing items of data is the table, or relation. A table consists of a variable number of rows. There is no predefined order in which the rows of a table are stored. Each row consists of a fixed number of columns, or fields. Columns are identified by column names and are defined to contain values of a specific data type (e.g., character, number, date). A particular column or group of columns is defined as a unique index for the table. This means that the values of those columns will be unique for every row in the table. There may also be other columns that are indexed but do not have to be unique across all rows. Certain columns exist only to define the relationship of a given row to rows in other tables. If the values in a column from one table are drawn from the same domain as the values in a column from another table, the data in the two tables are related where rows in each table share a common value. This basic organization is illustrated in Figure 1-1.

Figure 1-1 contains two tables, PROJECT and PROJ_SUB. The row in the PROJECT table for the project named XYZ is related, via common values in the project number columns (PROJ_NO), to a group of rows in the PROJ_SUB table representing XYZ's subsystems. The primary key in the PROJECT table is the project name column (PROJ_NAME), while the primary key in the PROJ_SUB table is the combination of the project number (PROJ_NO) and the subsystem prefix (SUB_PRE) columns. For more details, Reference 6 provides a good overview of relational database concepts. For ORACLE-specific information, References 4 and 5 provide an overview of the ORACLE RDBMS as well as a detailed description of the ORACLE structured query language (SQL).

Previous versions of this document mentioned that the SEL database contained clusters. The SEL database no longer has any clusters and all reference have been removed.

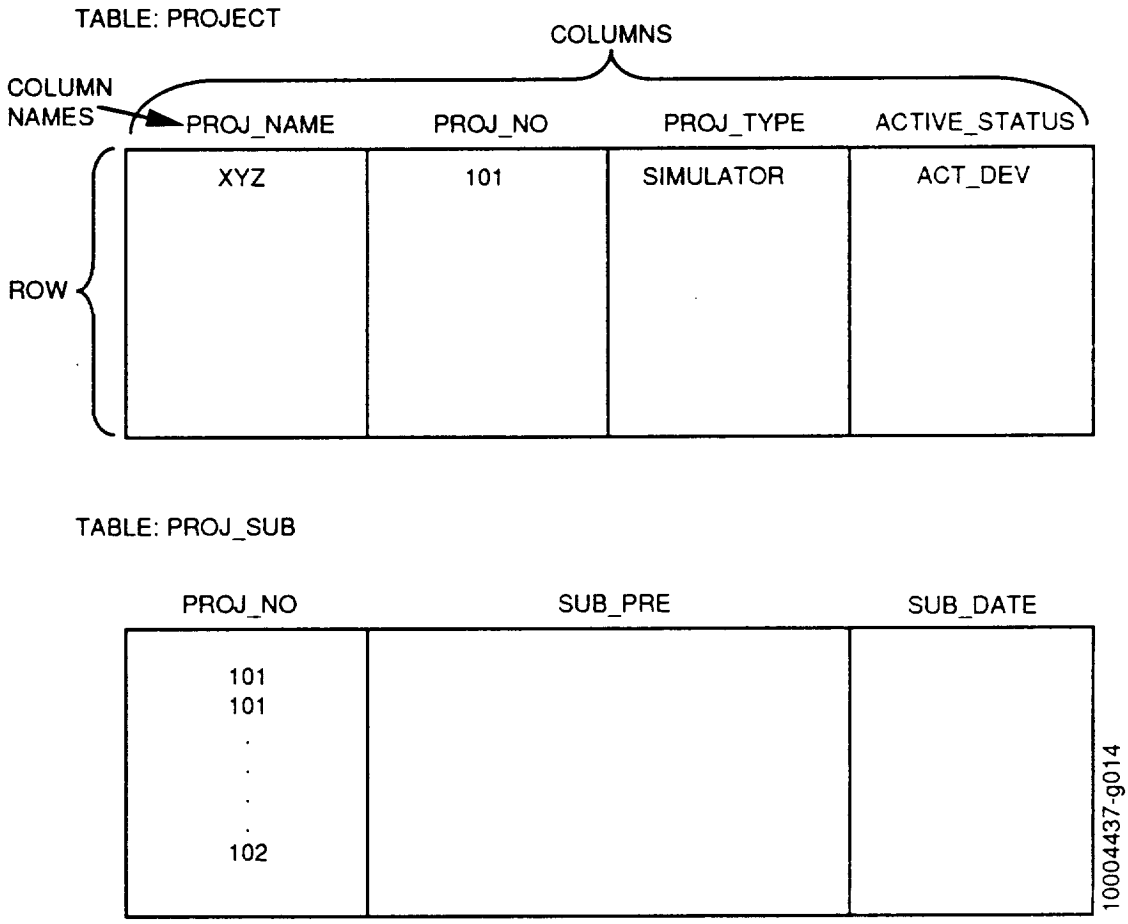


Figure 1-1. Basic Relational Database Organization

SECTION 2—A CONCEPTUAL VIEW OF SEL DATA

This section presents an overview of the types of software engineering data that are stored in the SEL database from a conceptual point of view. The fundamental entity about which SEL data are collected and stored is the project. Project data compose the bulk of the data in the database and are presented in Section 2.1. A relatively small portion of the database is allocated to the storage of support data, such as computer and personnel names. These data, which are not associated exclusively with individual projects, are referred to as project-independent data throughout this document. Section 2.2 contains detailed descriptions of these data. The data elements described in this section are tagged with the reference identifiers used in Sections 3 and 4.

Figure 2-1 shows the major data items that make up both the project data and the project-independent data. This conceptual view of the data is later mapped into the logical view of the SEL database discussed in Section 4. In the figure, data items flagged with asterisks are collected both during development and maintenance stages. The rest are collected only in projects' development stages.

2.1 PROJECT DATA

Software development in the area of flight dynamics at GSFC is performed in distinct units referred to by the SEL as projects. A project exists for a specified period of time that spans the life of a particular software product. The life of a project comprises two primary stages: the development stage and the operations and maintenance stage. The majority of the data collected by the SEL cover the development stage of the lifespan, although some data, such as resources and changes, are also collected during the maintenance stage. The following sections describe data types that characterize the development stage as well as data types that are captured during the maintenance stage. In addition, each project has associated with it the following general information that defines and identifies the project:

P1 Name of the project; a unique identifier distinguishing it from other projects

P2 Type of project; indicator used to describe the nature of the application and to identify projects with similar applications for the purpose of comparison

P3 Current status of the project; whether it is in the development stage or the maintenance stage or whether its life cycle has been completed or discontinued

P4 Miscellaneous descriptive information; this is optional data and may include any of the following:

- Project's full name
- Contacts for the project

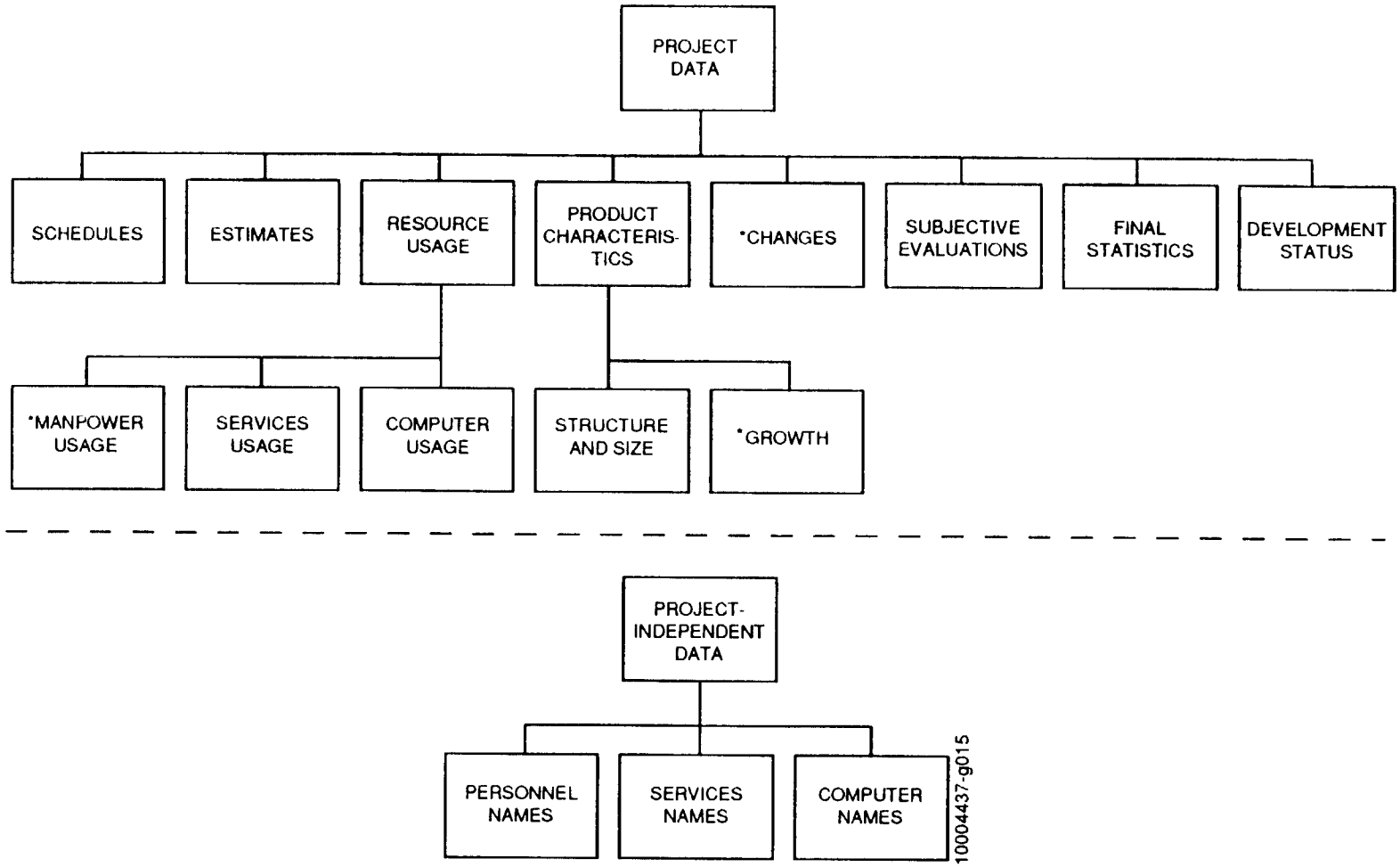


Figure 2-1. Conceptual View of SEL Data

- Language(s) used in a project
- Computer on which project is being developed and operated
- Computer accounts to be monitored by the SEL
- Project task numbers and corresponding years
- SEL forms collected for the project
- General notes on project or data peculiarities
- Name of the project controlled source library
- Tools used for collecting project growth data
- Project closeout status
- Types of data that are currently stored in the database for the project

2.1.1 Schedules

Project schedules divide the lifespan of a project into a series of nonoverlapping, contiguous time periods referred to by the SEL as phases. During the development stage, the phases correspond closely to the primary type of development activity being performed at any given time. The transition from one phase to the next is signaled by project milestones, such as the critical design review (CDR). The schedules stored in the database are supplied by personnel involved in managing the projects being monitored. An initial schedule is submitted at the start of the project and updated every 6 to 8 weeks thereafter until the completion of the project's development stage. All schedules submitted are stored in the database along with their submission dates to provide a historical trace of schedule changes. Schedule data exist in sets that include the following:

P1	Project name
P5	Date on which the schedule was recorded
P6	Requirements definition phase start and end dates
P7	Design phase start and end dates
P8	Implementation (code and test) phase start and end dates
P9	System test phase start and end dates
P10	Acceptance test phase start and end dates
P11	Cleanup phase start and end dates
P12	Maintenance stage start and end dates (not collected on current Project Estimates Form (PEF), but data exist for some projects)

Phase dates are subject to certain constraints, such as the requirement that they always fall on a Saturday. Also, depending upon the life-cycle model followed, the size and level of formality of the project, and the SEL's research needs, some of the phase dates may not be supplied for particular projects. Reference 1 presents a more thorough discussion of the SEL definition of phase dates and the constraints to which they must adhere.

2.1.2 Estimates

At various points in the life of a project, estimates are made of certain project characteristics whose actual values do not become available until the end of the development phase. These projections are made as part of the process of planning the project and monitoring its progress. As the project proceeds, the estimates are updated regularly to reflect such factors as system growth and changes in staffing patterns. Thus, toward the end of the development phase, the at-completion estimates converge on the actual final project characteristics. The sets of estimates collected by the SEL and stored in the database include the following:

P1 Project name

P13 Date on which the set of estimates was recorded

P14 Number of subsystems in the software product

P15 Number of components in the software product

P16 Total source lines of code (SLOC) in the software product

P17 Total SLOC for all reused components in the software product

P18 Total SLOC for all modified components in the software product

P19 Total SLOC for all new components in the software product

P20 Programmer hours spent on the project

P21 Management hours spent on the project

P22 Services hours spent on the project

The terms "subsystem" and "component," used above and elsewhere in this document, have specific definitions in the SEL environment. In general, subsystems are a mutually exclusive partitioning of the components that constitute a software system. Components, or modules, are individual routines that are maintained in separate files. (See Reference 1 for a more detailed description of these concepts.)

The SLOC estimates refer to total lines of source code, including executable and nonexecutable statements, comments, and blank lines. The total lines estimate is expected to be the sum of the old, modified, and new lines estimates. The programmer hours estimate is a projection of the total technical effort to be spent on the project. Similarly, the management hours

estimate is a projection of the total hours to be charged to project management. The services hours estimate is a projection of the hours to be spent by support personnel on the project. This includes secretaries, technical editors, word processors, couriers, and project control personnel.

2.1.3 Resource Use

Throughout the development stage of a project, the use of personnel and computer resources is measured and stored on a weekly basis. However, only the personnel resource use is measured when a project starts its maintenance phase.

2.1.3.1 MANPOWER

Development

Each week, the staff resources expended on a given project are recorded and stored in the database. Hours are stored for each person who does technical work or directly manages the project during the particular week in question. These hours are categorized by the type of development activity being performed. Thus, for any given project, week, and programmer, the following data are stored:

P1 Project name

P23 Week ending date; this date is always a Friday

P24 Personnel name; name of the person performing technical or direct management work on the project

P25 Predesign hours; hours worked on the project before commencement of actual design work (requirements definition, requirements analysis, etc.)

P26 Create design hours; hours spent performing software design activities (creating structure charts, writing program design language (PDL), etc.)

P27 Read and review design hours; hours spent reading and reviewing design materials (peer reviews, design walkthroughs, etc.)

P28 Write code hours; hours spent developing source code from design materials (coding at desk, entering code at terminal, etc.)

P29 Read and review code hours; hours spent reading code for any purpose except isolation of errors (peer review, code walkthroughs, desk checks, etc.)

P30 Test code unit hours; hours spent testing individual code units (planning and executing test cases, writing test drivers and stubs, etc.)

P31 Debug hours; hours spent isolating errors and planning corrections (does not include actually correcting errors)

P32 Integration test hours; hours spent planning tests that integrate system components (writing and executing system tests, etc.)

P33 Acceptance test hours; hours spent running and supporting acceptance testing of the software

P34 Other hours; hours that do not fall into any of the above activities (management, training, documentation, etc.)

The hours that are recorded in the various activities for a given programmer during a given week add up to the total hours worked on the project during that week by that programmer. Manpower hours are recorded to the nearest tenth of an hour. For projects that began before June 1987, the activity hour items P25 through P34 may be further classified by being associated with the subsystem on which the work was performed. In this case, the sum of the hours recorded in the various activities and associated with particular subsystems plus the hours charged to various activities and not associated with particular subsystems represents the total hours worked during that week by that programmer. An example of the latter case is as follows:

Programmer: J. Doe	Week ending: 30–Nov–87
Integration test hours (P32) for subsystem XYZ:	5.0
Integration test hours (P32) for subsystem ABC:	10.0
Write code hours (P28) for subsystem ABC:	15.0
Other hours (P34) (no subsystem):	10.0
Total hours worked:	40.0

In addition to and independent of these activity hours, programmer hours for the week are collected for the following activities:

P35 Rework hours; hours spent reworking any part of the system due to errors or other unplanned changes (includes rework of code, design, testing, and all hours spent debugging)

P36 Enhancing, refining, and optimizing hours; hours spent improving efficiency or clarity of design, code, or documentation (not due to unplanned changes)

P37 Documenting hours; hours spent creating any form of documentation on the system (system descriptions, user's guides, in-line comments, etc.)

P38 Reuse hours; hours spent attempting to reuse components of this or other systems

The hours recorded in the above categories do not adhere to the constraint that their sum must represent the total hours worked by a given programmer during a given week.

Certain projects in the database were developed using a cleanroom methodology. Consequently, the types of development activities recorded for these projects are different from

those mentioned above. However, staff resources expended on these projects are still recorded weekly and hours are still stored for each person who does technical work or directly manages the project. The following are the data stored for projects using a cleanroom methodology:

P1 Project name

P23 Week ending date; this date is always a Friday

P24 Personnel name; name of the person performing technical or management work on the project

P157 Predesign hours; hours worked on the project prior to the actual design (such as requirement analysis, etc.)

P158 Pretest hours; hours worked on developing test plans and building test environments (compiling components, building libraries, defining input, etc.)

P159 Create design hours; hours spent developing system, subsystems, or components design (state machine representation, data and stepwise refinement, PDL, etc.)

P160 Verify and review design; hours spent verifying and reviewing design in a group, including design meetings, formal and informal reviews, or walkthroughs

P161 Write code hours; hours spent coding system components (coding at desk, entering code at terminal, etc.)

P162 Read and review code hours; hours spent reading code for any purpose other than isolation of errors (code verification)

P163 Independent test hours; hours spent generating and executing tests of system components (by independent tester)

P164 Response to software failure report (SFR) hours; hours spent resolving a tester-reported problem (isolating a reported problem and developing a solution)

P165 Acceptance test hours; hours spent running and supporting acceptance testing of the software

P166 Other hours; hours spent on activities not covered above (management, meetings, training, documentation, etc.)

In addition to and independent of these cleanroom development activity hours, any weekly programmer hours spent understanding the methodology are captured under the following category:

P167 Methodology Understanding and Discussion; hours spent learning, discussing, or receiving training in cleanroom-related methods and techniques

Reference 1 presents a more detailed discussion of the various activities that categorize manpower effort hours.

Maintenance

When a project completes its development cycle and starts its maintenance stage, the use of personnel resources is also measured and stored. Each week, the regular maintainers' resources expended on a given maintenance project are recorded. Hours are stored for each person who does technical work or directly manages the project. The hours are categorized by both the class of maintenance and by the type of activity being performed. Thus, for any given maintenance project, the following data are stored:

P1 Project name

P23 Week ending date; this date is always a Friday

P24 Personnel name; name of the person performing technical or management work on the maintenance project

P168 Correction class hours; hours worked on all maintenance associated with a system failure

P169 Enhancement class hours; hours spent on all maintenance associated with modifying the system due to a requirements change

P170 Adaptation class hours; hours spent on all maintenance associated with modifying a system to adapt a change in hardware, software, or environment characteristics

P171 Other class hours; hours spent on all maintenance that do not fall into any of the above classes (management, meetings, etc.)

P172 Isolation activity hours; hours spent on understanding the failure or request for enhancement or adaptation

P173 Change design activity hours; hours spent on redesigning the system

P174 Implementation activity hours; hours spent on changing the system to complete the necessary change (hours include changing not only the code, but the associated documentation as well)

P175 Unit or system test activity hours; hours spent on testing the changed or added component

P176 Acceptance or benchmark test activity hours; hours spent on acceptance or benchmark testing

P177 Other activity hours; hours that do not fall into any of the above activities (management, meetings, etc.)

2.1.3.2 SERVICES

Each week during the development stage of a project, services hours are recorded and stored in the database. These are hours spent by support personnel who are not directly involved in

the technical aspects of the project. The categories of services hours recorded each week for a given project are as follows:

P1 Project name

P23 Week ending date; this date is always a Friday

P39 Technical publications hours; hours spent by technical editors, word processors, graphic artists, etc., in preparing technical documentation for the project

P40 Secretary hours; hours spent by secretarial personnel in direct support of the project

P41 Librarians; hours spent by data librarians in support of the project, e.g., data entry, tape generation (not collected on current Service/Products Form (SPF) but data exist for some old projects)

P42 Project management; hours spent by persons performing management activities in support of the project, but who are not directly responsible for the project's management

P43 Other; hours spent in support of the project by personnel who do not qualify in one of the support service categories above

Service hours are not recorded for individuals. Rather, the sum of the hours reported by all persons performing a particular support activity during a given week is recorded.

2.1.3.3 COMPUTER

Computer resources are the third type of resource data recorded and stored in the database on a weekly basis. During the portion of the development stage when programmers are using computer resources to create the resulting software product, the number of computer runs and central processing unit (CPU) hours used are monitored. If different portions of the development effort are performed on different machines, hours and runs are recorded for each of them. Thus, for each week of a given project, the following computer resource data are stored:

P1 Project name

P23 Week ending date; this date is always a Friday

and for each computer being used at the current time:

P44 Computer name; name uniquely identifying the development computer

P45 CPU hours used

P46 Number of runs executed

The number of runs recorded is measured as either the number of interactive logons by project members, the number of batch jobs submitted by project members, or both. On some development computers, the accounting reports used for obtaining the resource data show separate CPU time and number of run statistics for interactive sessions and batch jobs. In these cases, the two are recorded separately under distinct computer names. On other machines, the accounting reports show total CPU time and number of runs without distinguishing between batch jobs and interactive sessions. In these cases, only the single combined figures are recorded.

2.1.4 Product Characteristics

A fourth class of project-related data characterizes the software product that is generated during the development stage. There are two primary types of product data: that which captures the static composition of the system at any given point in time, and that which captures the dynamic properties of system growth and change.

2.1.4.1 STRUCTURE AND SIZE

The static composition of the system is recorded as the system is produced. This consists of the partitioning of the system into subsystems and components, along with descriptive information about each. As mentioned earlier, the SEL defines subsystems as a mutually exclusive partitioning of the system components. For each subsystem in a project, the following data items are stored:

P1 Project name

P47 Subsystem prefix; mnemonic prefix used in naming components that belong to the subsystem

P48 Subsystem name; descriptive name describing the purpose of the subsystem

P49 Subsystem function; indicator used to describe the nature of the subsystem and also to identify similar subsystems for the purpose of comparison

P50 Date on which the subsystem information was recorded

Subsystem prefixes are unique within a given project. Each subsystem comprises multiple components. Components are defined as modules or routines that are maintained in separate files as individual configuration items. Each component is associated with exactly one subsystem. The following descriptive information is stored for each component of the system:

P24 Programmer name; name of programmer who created the component

P1 Project name

P47 Subsystem prefix; prefix identifying the subsystem to which the component belongs

P51 Component name; descriptive name used in identifying the component

P52 Component date; date on which the component information was recorded by the programmer

P53 Creation date; date on which the component first became part of the system configuration (i.e., was moved into the controlled source library)

P56 Origin; source of the component (i.e., old code, modified old code, new code)

P57 Difficulty; discrete rating on a scale of 1 (easiest) to 5 (most difficult) of the difficulty in creating the component

P58 Type; indicator used to classify components of similar nature for comparison

P59 Purpose; indicator of the component's purpose

2.1.4.2 GROWTH

Growth data recorded in the SEL database capture the dynamic nature of the evolving software product. These data are obtained by taking snapshots of the controlled source library of the project at regular intervals (weekly for development projects, monthly for maintenance projects). The data elements captured each week provide a historical perspective on system size through the development stage of the life cycle. The information recorded is as follows:

P1 Project name

P23 Week ending date; this data is always a Friday

P60 Lines of code; count of the total lines of code in the project's controlled source library

P61 Components; count of the number of components in the project's controlled source library

P62 Changes; count of the number of changes that have occurred in the project's controlled library (each time a new component is added to the library, it is counted as one change; each time a component is updated in the library, it is counted as another change)

2.1.5 Changes

Development

Detailed information is recorded in the database for each change that takes place in a project's configured software library (or libraries). A change is viewed by the SEL as an update to one or more system components for a particular specific purpose. Typical purposes for changes include correcting an error, improving the efficiency of a particular operation, or implementing an enhancement. The following data items are stored for each change:

P1 Project name

P63 Change number; number uniquely identifying each change in the database

P24 Programmer name; name of the programmer implementing the change

P65 Change date; date on which the change information was recorded

P66 Effort required to isolate the change; time spent determining what was necessary to make the change

P67 Effort required to implement the change; time spent actually designing, coding, and testing the change

P68 One component affected; flag indicating whether the change involved updating only one component

P69 Involved Ada; flag indicating whether the change resulted from using the Ada language

P70 Examined other components; flag indicating whether components other than those changed were examined when performing the change

P71 Parameters passed; flag indicating whether the change required awareness of data communicated between components

P72 Date change determined; date on which the need for the change was initially determined

P73 Date change completed; date on which the change was implemented into the system

P74 Number of components changed; count of the changed components

P75 Number of components examined; count of the components examined in the change process that were not changed themselves

P76 Change type; indicator used to classify changes by particular types

P77 Error source; indicator of the source of the error for changes where the change type (P76) is error correction

P78 Error class; indicator of the class of error for changes where the change type (P76) is error correction

P79 Commission error; for changes where the change type (P76) is error correction, flag indicating whether something incorrect was included in the code

P80 Omission error; for changes where the change type (P76) is error correction, flag indicating whether something was left out of the code

P81 Typographical error; flag indicating whether an error was typographical in nature for changes where the change type (P76) is error correction

P82 Ada documentation; flag indicating whether the Ada documentation clearly explained the features that contributed to an error (P76) attributed to the use of Ada (P69)

P83 Ada cause; indicator of the cause of an error (P76) attributed to the use of Ada (P69)

P84 Changed components; subsystem prefixes and names of the components that were changed

P85 Ada features; list of the Ada features that were involved in a change (P76) in which the use of Ada was a contributing factor (P69)

P86 Ada resources; list of resources used in resolving an Ada-related error (P69,P76)

P87 Ada tools; list of software tools used in resolving an Ada-related error (P69,P76)

Maintenance

Detailed information is also recorded for each change that takes place in a project's controlled library during the maintenance stage. The definition of change is the same as mentioned in the change (development) section. The following data items are stored for each change:

P1 Project name

P24 Programmer name; name of the programmer implementing the change

P65 Change date; date on which the change information was recorded

P178 Operational Software Modification Report (OSMR) number

P179 Change type; indicator used to classify changes by particular types

P180 Change cause; indicator used to classify the cause of a particular change

P181 Effort required to isolate the change; time spent determining what was necessary to make the change

P182 Effort required to implement the change; time spent actually designing, coding, and testing the change

P183 Changed object types; list of objects that have been changed as a result of this change

P184 Change characteristic; indicator used to classify the characteristic of this change

P185 Number of SLOC that have been newly added (the total SLOC includes blanks and comments)

P186 Number of SLOC that have been modified

P187 Number of SLOC that have been deleted

P188 Number of components that have been newly added

P189 Number of components that have been modified

P190 Number of components that have been deleted

P191 Number of the added components that are totally new

P192 Number of the added components that are totally reused

P193 Number of the added components that are reused with modifications

2.1.6 Subjective Evaluations

When a project completes its development stage, the retrospective subjective opinions of personnel involved in the management of the project are collected and stored in the database. This includes rating a set of project characteristics on a scale of 1 to 5 and indicating what software engineering tools were used on the project. Unless otherwise specified, the scale on the measures ranges from 1 = low to 5 = high. The subjective data items recorded are as follows:

P1 Project name

P88 Problem complexity

P89 Schedule constraints (loose = 1, tight = 5)

P90 Stability of requirements (unstable = 1, stable = 5)

P91 Quality of requirements

P92 Documentation requirements

P93 Rigor of requirements reviews

P94 Development team ability

P95 Development team application experience

P96 Development team environment experience

P97 Stability of development team (unstable = 1, stable = 5)

P98 Management performance

P99 Management application experience

P100 Stability of management team (unstable = 1, stable = 5)

P101 Project planning discipline

P102 Degree to which plans were followed

P103 Use of modern programming practices

P104 Discipline in formal communication

P105 Discipline in requirements methodology

P106 Discipline in design methodology

P107 Discipline in testing methodology

P108 List of tools used on project (not a numerical rating, but an actual list of tool names)

P109 Use of test plans

P110 Discipline in quality assurance

P111 Discipline in configuration management

P112 Access to development system

P113 Ratio of developers to terminals (low = 5, high = 1)

P114 Memory constraints

P115 System response time (poor = 1, very good = 5)

P116 Stability of hardware and support software

P117 Effectiveness of tools used

P118 Agreement of software with requirements

P119 Quality of software

P120 Quality of design

P121 Quality of documentation

P122 Timeliness of delivery

P123 Smoothness of acceptance testing

2.1.7 Final Statistics

When the development stage of a project is complete, the actual values of parameters that were estimated earlier and of additional parameters that were not estimated are recorded. In

addition, the project source code is run through a static analysis tool, and statistics are recorded for each component of the system. The data items that constitute final project statistics are as follows:

P1 Project name

P124 Date on which the final statistics were recorded

P125 Actual requirements definition phase start and end dates

P126 Actual design phase start and end dates

P127 Actual code and test (implementation) phase start and end dates

P128 Actual system test phase start and end dates

P129 Actual acceptance test phase start and end dates

P130 Actual cleanup phase start and end dates

P131 Maintenance stage start and end dates

P132 Total technical and management hours expended on the project

P133 Total service hours expended on the project

P134 Computer name

P135 CPU hours used

P136 Number of runs executed, for each computer used on the project

P137 Number of subsystems in the system

P138 Number of components in the system

P139 Number of changes made to system components

P140 Number of pages of documentation produced for the system

P141 Total SLOC for all components in the system

P142 Total SLOC for all components in the system that were classified as new

P143 Total SLOC for all components in the system that were classified as slightly modified

P213 Total SLOC for all components in the system that were classified as extensively modified

P144 Total SLOC for all components in the system that were reused from other systems without modification

P145 Total number of comment lines for all components in the system

P146 Total number of executable components in the system

P147 Total number of newly created executable components in the system

P148 Total number of executable components in the system that were obtained from other systems and slightly modified for this project

P214 Total number of executable components in the system that were obtained from other systems and extensively modified for this project

P149 Total number of executable components in the system that were reused from other systems without modification

P150 Total number of executable statements for all FORTRAN components in the system

P151 Total number of executable statements for all FORTRAN components in the system that were classified as new

P152 Total number of executable statements for all FORTRAN components in the system that were classified as slightly modified

P215 Total number of executable statements for all FORTRAN components in the system that were classified as extensively modified

P153 Total number of executable statements for all FORTRAN components in the system that were reused from other systems without modification

P216 Total number of statements for all components in the system

P217 Total number of statements for all components in the system that were classified as new

P218 Total number of statements for all components in the system that were classified as slightly modified

P219 Total number of statements for all components in the system that were classified as extensively modified

P220 Total number of statements for all components in the system that were reused from other systems without modification

and for each component in the system:

P154 Number of executable statements in the component (for FORTRAN components only)

P155 Number of SLOC in the component (includes comments and blank lines)

P156 Number of comment lines in the component (for FORTRAN or Ada components only; does not include blank lines)

P221 Number of statements in the component (for FORTRAN or Ada components only)

P222 Final origin category assigned to the component

2.1.8 Development Status Data

The status of active projects is monitored throughout project development and recorded in the SEL database. The data items are recorded on a biweekly basis for each active project. There are two types of development status data: target data and measurement data. The target data represent the goal or target value. The measurement data represent a value measuring the progress toward the target value. The following data items are stored:

P1 Project name

P23 Week ending date; this date is always a Friday

P24 Name of originator

P195 Total number of components to be designed

P196 Number of components designed as of the week ending date

P197 Total number of components to be coded

P198 Number of components coded as of the week ending date

P199 Total number of separate system tests planned

P200 Number of system tests executed at least one time

P201 Number of system tests passed

P202 Total system test runs, including reruns (not collected on current Development Status Form (DSF), but data exist for some projects)

P203 Total number of separate acceptance tests planned

P204 Number of acceptance tests executed at least one time

P205 Number of acceptance tests passed

P206 Total acceptance test runs, including reruns (not collected on current DSF, but data exist for some projects)

P207 Total number of discrepancies reported

P208 Total number of discrepancies resolved

P209 Total number of specification modifications received

P210 Total number of specification modifications completed

P211 Total number of requirements questions submitted

P212 Total number of requirements questions answered by analysts

2.2 PROJECT-INDEPENDENT DATA

This section describes two types of data stored in the database that represent real-world entities, yet are not directly related to a particular project, as were the items in the previous section. The data stored about these items are not extensive. Rather, their primary function is to identify specific instances of resources when recording project data.

2.2.1 People and Services

The first class of support entities consists of people and services. Each person for whom data are recorded is represented in the database by the following data items:

M1 Form name; abbreviated version of the person's name used on data collection forms (see Section 3)

M2 Full name; person's complete first and last name

M3 Entry date; date on which personnel information was entered into the database

Service personnel are stored in the database generically; that is, the same information listed above is stored as only one generic entry for a given class of service personnel. Thus, for example, the personnel entry for secretary refers collectively to anyone performing secretarial work on a monitored project.

2.2.2 Computer

The other class of support entities is computers. Each computer for which resource hours and runs are recorded is represented in the database by the following data items:

M4 CPU name; abbreviated version of the computer name used on data collection forms (see Section 3)

M5 Computer full name; longer, more descriptive name for the computer

SECTION 3—SEL DATA FROM A DATA COLLECTION VIEWPOINT

This section describes the data collection forms in their role as sources for the data items described in Section 2. Many data items entered on the forms map directly to items described in Section 2. Other items (e.g., form numbers) are unique to the data collection process and therefore do not appear in Section 2. This section maps the software engineering items in Section 2 to their sources on data collection forms and describes the data items that are peculiar to the data collection process.

The following subsections present descriptions for the SEL data collection forms. The data items described are tagged with reference identifiers corresponding to the identifiers in the forms that are presented in Appendix C. The identifiers are also used as cross references in the SEL database access paths (Table 4-4 in Section 4). If an item maps directly to an item in Section 2, the description consists of the item name followed by the Section 2 identifier for that item (in parentheses). Otherwise, a more complete description is presented.

3.1 DATA COLLECTION FORMS

3.1.1 Schedule and Estimates Forms

The PEF (Figure C-8 in Appendix C) provides periodic estimates of the development process and the software product and estimates of the project schedule. The estimates of the development process consist of staffing projections. The estimates of the software product involve various estimates of the size of the delivered software. The schedule information consists of a set of dates on which the various life-cycle phases of the project are scheduled to start, along with a projected project end date. These estimates reflect the project size and resource expenditure as of the completion of the cleanup phase.

The PEF is completed by the project leader. It is submitted at the initial entry of the project into the database and every 6 to 8 weeks thereafter through the development life cycle. The PEF data fields are described below. Note that the phase date fields contain the start dates of each of the listed life-cycle phases that apply to the project. The end date for a given phase is the next phase start date entered on the form, or the project end date if there are no start dates for subsequent phases.

PEF Fields

D1 Project name (P1)

D2 Form date (P5, P13)

D3 Requirements; estimated requirements definition phase start date (P6)

D4 Design; estimated design phase start date (P7)

D5 Implementation; estimated implementation (code and test) phase start date (P8)

D6 System test; estimated system test phase start date (P9)

D7 Acceptance test; estimated acceptance test phase start date (P10)

D8 Cleanup; estimated cleanup phase start date (P11)

D10 Project end; estimated project end date

D11 Programmer hours (P20)

D12 Management hours (P21)

D13 Services hours (P22)

D14 Number of subsystems (P14)

D15 Number of components (P15)

D16 Total SLOC (P16)

D17 Total SLOC for all new Components (P19)

D18 Total SLOC for all modified components (P18)

D19 Total SLOC for all reused components (P17)

D20 PEF form number; unique identifier distinguishing this form from other PEFs

3.1.2 Weekly Rate Data Forms

The Personnel Resource Form (PRF) or the Cleanroom Personnel Resource Form (CLPRF) and the SPF provide weekly rate information for the projects in their development stage. The SPF is also used to provide monthly growth rate information for projects in the maintenance stage. The Weekly Maintenance Effort Form (WMEF) provides weekly rate information when a project starts its maintenance stage. The PRF and CLPRF (Figures C-5 and C-6), capture the actual technical/management expenditure history on the project. These forms also contain information on the type of activity on which the manpower hours were spent during the week. A separate section of the forms is used to record hours spent performing specific activities that are of current interest to the SEL.

The PRF is used to capture personnel hours for most of the SEL-monitored projects. It is submitted by every person performing either technical or management activities on the project. This form is completed every Friday for the duration of the project development life cycle.

PRF Fields

D21 Personnel name (P24)

D1 Project name (P1)

D22 Week ending date (P23)

D23 Predesign hours (P25)

D24 Create design hours (P26)

D25 Read/review design hours (P27)

D26 Write code hours (P28)

D27 Read/review code hours (P29)

D28 Test code unit hours (P30)

D29 Debugging hours (P31)

D30 Integration test hours (P32)

D31 Acceptance test hours (P33)

D32 Other hours (P34)

D33 Rework hours (P35)

D34 Enhancing/refining/optimizing hours (P36)

D35 Documenting hours (P37)

D36 Reuse hours (P38)

D37 PRF form number; unique identifier distinguishing this form from other PRFs

The CLPRF is submitted by personnel who work on projects that use cleanroom methodology to do software development. This form is submitted by every person performing either technical or management activities on the project. This form, like the PRF, is completed every Friday for the duration of the project development life cycle.

CLPRF Fields

D21 Personnel name (P24)

D1 Project name (P1)

D22 Week ending date (P23)

D199 Predesign hours (P157)

D200 Pretest hours (P158)

D201 Create design hours (P159)

D202 Verify/review design hours (P160)

D203 Write code hours (P161)

D204 Read/review code hours (P162)

D205 Independent test hours (P163)

D206 Response to SFR hours (P164)

D207 Acceptance test hours (P165)

D208 Other hours (P166)

D209 Methodology understanding/discussion (P167)

D210 CLPRF form number; unique identifier distinguishing this form from other CLPRFs

The WMEF (Figure C-14) is submitted by every person performing either technical or management activities on a maintenance project. The form is completed every Friday for the duration of the project's maintenance phase. In the WMEF, the activity hours are categorized as class of maintenance hours and as maintenance activity hours. The sum of the class of maintenance hours recorded in Section B is equal to the total hours provided in Section A of the form. The sum of the maintenance activities hours of Section C is also equal to the total hours provided in Section A. The users can choose one of the two categories to calculate the total maintenance manpower hours for the project.

WMEF Fields

D21 Personnel name (P24)

D1 Project name (P1)

D22 Week ending date (P23)

D151 Correction hours (P168)

D152 Enhancement hours (P169)

D153 Adaptation hours (P170)

D154 Other hours (P171)

D155 Isolation hours (P172)

D156 Change design hours (P173)

D157 Implementation hours (P174)

D158 Unit test/system test hours (P175)

D159 Acceptance/benchmark test hours (P176)

D160 Other hours (P177)

D161 WMEF form number; unique identifier distinguishing this form from other WMEFs

The SPF (Figure C-11) measures resource expenditure by support personnel, and computer resource utilization, and is used to create a historical record of product growth over the course of the project. The SPF is completed by SEL data collection personnel. The form contains three distinct types of data; the growth history data are obtained by running growth history monitoring programs on the Flight Dynamics Facility (FDF) mainframes (two ES/9000s and two NAS 8063s) and the STL VAX Cluster (8820, 11/780, and Micro VAX 3100). The computer information is taken from computer accounting reports from these computers. Services hours are obtained from task accounting reports. This form is submitted every week in which support service or computer resources are used or in which product growth data are available. This form is submitted monthly for all maintenance projects for which growth data is being monitored.

SPF Fields

D1 Project name (P1)

D22 Week ending date (P23)

D38 Computer name (P44)

D39 CPU hours (P45)

D40 Number of runs (P46)

D41 Number of components (P61)

D42 Number of changes (P62)

D43 Lines of code (P60)

D44 Technical publications hours (P39)

D45 Secretary hours (P40)

D47 Project management hours (P42)

D48 Other hours (P43)

D49 SPF form number; unique identifier distinguishing this form from other SPFs

3.1.3 Product Data Forms

The Subsystem Information Form (SIF), the Component Origination Form (COF), and the Change Report Form (CRF) provide product data information for the project during its development stage. The Maintenance Change Report Form (MCRF) provides product data information for the project when it moves into its maintenance stage.

The SIF (Figure C-13) contains information about the high-level partitioning of the system into subsystems. A subsystem prefix, a descriptive name, and a subsystem function should be

specified for each subsystem. The SIF is completed by the project leader. A form is submitted at the time of the preliminary design review (PDR) and any time thereafter when a new subsystem is introduced into the design of the system.

SIF Fields

D1 Project name (P1)

D2 Form date (P50)

D50 Subsystem prefix (P47)

D51 Subsystem name (P48)

D52 Subsystem function (P49)

The COF (Figure C-2) records information about a component in the system. Some of the information collected are the origin of the component, difficulty of developing the component, type of component, and purpose of component. The COF is completed by personnel who code new system components, modify old components for reuse, or transfer reused components to the project's controlled library. A form is completed for each component in the system at the time when the component is moved into the project controlled source library.

COF Fields

D21 Programmer Name (P24)

D1 Project Name (P1)

D2 Form Date (P52)

D50 Subsystems Prefix (P47)

D53 Component name (P51)

D54 Date entered into controlled library (P53)

D55 Relative difficulty of developing component (P57)

D56 Origin (P56)

D57 Type of component (P58)

D58 Purpose of executable component (P59)

D59 COF form number; unique identifier distinguishing this form from other COFs

The CRF (Figure C-1) contains information about the type of change that was made, the components that were changed, error information if applicable, and Ada-specific informa-

tion if applicable. The CRF is completed by personnel who implement changes to the system that involve modifying components in the project's controlled source library. A form is submitted for each change to the system at the time the changed components are updated in the project's controlled source library.

CRF Fields

D21 Programmer name (P24)

D1 Project name (P1)

D2 Form date (P65)

D50 Subsystem prefixes of components changes (P84)

D53 Names of components changed (P84)

D63 Date on which need for change was determined (P72)

D64 Date change was completed (P73)

D65 Effort to isolate change (P66)

D66 Effort to implement change (P67)

D67 Type of change (P76)

D68 Change to one component (P68)

D69 Look at any other components (P70)

D70 Aware of parameters (P71)

D71 Source of error (P77)

D72 Class of error (P78)

D73 Omission error (P80)

D74 Commission error (P79)

D75 Transcription error (P81)

D76 Did Ada contribute to the change (P69)

D77 Ada features involved (P85)

D78 Documentation understandable (P82)

D79 Which statement best describes the cause of the Ada error (P83)

D80 Which resources provided the information needed to correct the error (P86)

D81 Which tools provided aided in correction of the error (P87)

D82 CRF form number (P63)

The MCRF (Figure C-4) contains information about the type of change that was made to the components in a project's maintenance controlled library. This form is submitted whenever the maintenance programmer has completed the work associated with a particular OSMR.

MCRF Fields

D21 Programmer name (P24)

D162 OSMR number (P178)

D1 Project name (P1)

D2 Form date (P65)

D163 Type of change (P179)

D164 Cause of change (P180)

D165 Effort to isolate change (P181)

D166 Effort to implement change (P182)

D167 Changed objects (P183)

D168 Change characteristic (P184)

D169 Number of lines of code added (P185)

D170 Number of lines of code changed (P186)

D171 Number of lines of code deleted (P187)

D172 Number of components added (P188)

D173 Number of components changed (P189)

D174 Number of components deleted (P190)

D175 Number of added components that are totally new (P191)

D176 Number of added components that are totally reused (P192)

D177 Number of added components that are reused with modifications (P193)

D178 MCRF form number; unique identifier distinguishing this form from other MCRFs

3.1.4 Project Development Completion Forms

The Project Completion Statistics Form (PCSF) and the Subjective Evaluation Form (SEF) provide project completion information for projects that have completed development and have been delivered to maintenance and operations. The PCSF (Figure C-7) is used to record the final development statistics for the project. This information includes the actual project resource expenditures, project schedule, and the software product size.

The PCSF is completed by SEL personnel and is verified by the project leader. It is completed during "closeout", a process of project data validation and verification. The PCSF data fields are described below. Note that, as in the PEF, the phase date fields contain the start dates of each of the listed life-cycle phases that apply to the project. The end date for a given phase is the next phase start date entered on the form, or the project end date if there are no start dates for subsequent phases.

PCSF Fields

D1 Project name (P1)

D2 Form date (P124)

D84 Requirements; actual requirements definition phase start date (P125)

D85 Design; actual design phase start date (P126)

D86 Implementation; actual implementation (code and test) phase start date (P127)

D87 System test; actual system test phase start date (P128)

D88 Acceptance test; actual acceptance test phase start date (P129)

D89 Cleanup; actual cleanup phase start date (P130)

D90 Maintenance; actual maintenance stage start date (P131)

D91 Project end; actual project end date

D92 Technical and management hours (P132)

D93 Services hours (P133)

D38 Computer name (P134)

D94 CPU hours (P135)

D95 Number of runs (P136)

D96 Number of subsystems (P137)

D97 Number of components (P138)

D98 Number of changes (P139)

D99 Pages of documentation (P140)

D100 Total SLOC (P141)

D101 Total SLOC for all new components (P142)

D102 Total SLOC for all slightly modified components (P143)

D211 Total SLOC for all extensively modified components (P213)

D103 Total SLOC for all old components (reused from other systems without modification) (P144)

D104 Comments (P145)

D105 Total executable components (P146)

D106 Total new executable components (P147)

D107 Total slightly modified executable components (P148)

D212 Total extensively modified executable components (P214)

D108 Total old executable components (reused from other systems without modification) (P149)

D109 Total executable statements for all FORTRAN components (P150)

D110 Total executable statements for all new FORTRAN components (P151)

D111 Total executable statements for all slightly modified FORTRAN components (P152)

D213 Total executable statements for all extensively modified FORTRAN components (P215)

D112 Total executable statements for all old FORTRAN components (reused from other systems without modification) (P153)

D214 Total statements (P216)

D215 Total statements for all new components (P217)

D216 Total statements for all slightly modified components (P218)

D217 Total statements for all extensively modified components (P219)

D218 Total statements for all old components (reused from other systems without modification) (P220)

D113 PCSF form number; unique identifier distinguishing this form from other PCSFs

The SEF (Figure C-12) consists of subjective perceptions of persons who were involved in managing the project with respect to such factors as the use of methodologies, the development environment, and the complexity of the problem. The SEF is completed by the project leader and selected personnel involved in managing the project. The responses from each of the completed forms are combined and reported on one form. The SEF is submitted when the final system products have been delivered (end of cleanup phase).

SEF Fields

D1 Project name (P1)

D2 Form date (P13)

D114 Problem difficulty or complexity (P88)

D115 Tightness of schedule constraints (P89)

D116 Stability of requirements (P90)

D117 Quality of specification documents (P91)

D118 Requirements for documentation (P92)

D119 Rigor of formal reviews (P93)

D120 Ability of development team (P94)

D121 Development team experience with application (P95)

D122 Development team experience with environment (P96)

D123 Stability of development team composition (P97)

D124 Project management performance (P98)

D125 Project management experience (P99)

D126 Stability of project management team (P100)

D127 Project planning discipline (P101)

D128 Degree project plans followed (P102)

D129 Modern programming practices (P103)

D130 Disciplined specification modification and question tracking (P104)

D131 Use of requirements analysis methodology (P105)

D132 Use of disciplined design methodology (P106)

D133 Use of disciplined testing methodology (P107)

D134 Use of tools (P108)

D135 Use of test plans (P109)

D136 Use of quality assurance procedures (P110)

D137 Use of configuration management procedures (P111)

D138 Degree of access to development system (P112)

D139 Programmers per terminal (P113)

D140 Development machine resource constraints (P114)

D141 System response time (P115)

D142 System hardware and support software stability (P116)

D143 Software tool effectiveness (P117)

D144 Delivered software supports requirements (P118)

D145 Quality of delivered software (P119)

D146 Quality of design present in delivered software (P120)

D147 Quality and completeness of software documentation (P121)

D148 Timely software delivery (P122)

D149 Smoothness of acceptance testing (P123)

D150 SEF form number; unique identifier distinguishing this form from other SEFs

3.1.5 Project Data Forms

The Project Startup Form (PSF) and Project Messages Form (PMF) are used to record miscellaneous descriptive information about a project. Both forms are completed by SEL personnel with information provided by the project leader.

The PSF (Figure C-10) is completed only once at project startup. The PSF information is obtained at the project startup meeting between SEL personnel and the project leader. The PSF data are stored as project messages.

PSF Fields

D1 Project name (P1)

D2 Form date

D60 Project type (P2)

D61 Project message type; NOTE_TYPEs of COMPACCTS, COMPSYS, CONTACTS, FORMSCOL, GENMESS, LANGUAGES, PROJNAME, and TASKNO (P4)

D62 Project message (P4)

The PMF (Figure C-9) captures general notes about a project, unique characteristics of the methodologies used, or peculiarities about the project's data. A PMF can be completed any time SEL personnel or the project leader feel that something about the project should be documented. A general message is always entered during project closeout.

PMF Fields

D1 Project name (P1)

D2 Form date

D61 Project message type; NOTE_TYPE of GENMESS (P4)

D62 Project message (P4)

3.1.6 Project Development Status Forms

The DSF provides project development status information for active projects. The DSF, (Figure C-3) is used to record such project status information as the number of components designed and coded and the number of tests performed. The DSF is completed on a bi-weekly basis by the project leaders of all active projects.

DSF Fields

D21 Name of originator (P24)

D1 Project name (P1)

D22 Week ending date; this date is always a Friday (P23)

D180 Total number of components to be designed (P195)

D181 Number of components designed as of the week ending date (P196)

D182 Total number of components to be coded (P197)

D183 Number of components coded as of the week ending date (P198)

D184 Total number of separate system tests planned (P199)

D185 Number of system tests executed at least one time (P200)

D186 Number of system tests passed (P201)

D188 Total number of separate acceptance tests planned (P203)

D189 Number of acceptance tests executed at least one time (P204)

D190 Number of acceptance tests passed (P205)

D192 Total number of discrepancies reported (P207)

D193 Total number of discrepancies resolved (P208)

D194 Total number of specification modifications received (P209)

D195 Total number of specification modifications completed (P210)

D196 Total number of requirements questions submitted to analysts (P211)

D197 Total number of requirements questions answered by analysts (P212)

D198 DSF form number; unique identifier distinguishing this form from other DSFs

SECTION 4—A LOGICAL VIEW OF THE SEL DATABASE

This section presents the logical schema of the SEL database. The introduction to relational databases in Section 1, together with the table descriptions in the following sections, allow the reader to understand where the data items described in Sections 2 and 3 may be found in the database. This section also presents some additional information about the way the data are stored and describes the tables containing database support data. These latter discussions are intended for the reader who needs to understand the database at a deeper level, such as a database maintenance programmer.

Section 4.1 defines each table in the SEL database. Section 4.2 describes how the tables are related to one another and constraints that are imposed on the tables by the semantics of the SEL data. Section 4.3 maps the data items as defined conceptually in Sections 2 and 3 to each item's location in a database table. This section also describes the access path to follow to reach each end data item.

In addition to the tables in the SEL database on the VAX, there are tables on the personal computer (PC) that are used for storing and maintaining DSF data. Since the DSF data are entered and quality assured by using the Database Access Manager for the SEL-PC (DAMSEL-PC) system, tables for storing DSF data are replicated on the PC. Some additional tables also exist on the PC to store validation data downloaded from the VAX database. This information is presented in Table 4-2 in a separate PC section. Tables for the VAX DSF data are described, along with others, both in Tables 4-1 and 4-2.

4.1 DATABASE TABLE AND VIEW DEFINITIONS

The SEL database contains a total of 78 base tables (relations) and 51 views. Base tables are defined independently of other tables in the sense that no base table is completely derivable from any other base table. On the other hand, views are virtual tables that are completely derived from base tables and contain no data of their own. With some restrictions, they can be treated as base tables. In the SEL database environment, views are used to provide users or application programmers with a more convenient way to access data items that spread across more than one base table. Tables 4-1 and 4-2 both present tables and views in the database and their component fields. Table 4-1 contains only 40 tables and 5 views (on the VAX), and is intended for all database users.

Table 4-2 contains additional tables and views that are mainly used for data entry, system maintenance, and project closeout, and are not relevant to general users. Table 4-1 presents the following information for each table and view included:

- Table or view name and a brief description of the data it contains
- For each column included in the table or view:

- Column name; an underlined column name is the primary key for accessing any table row. If multiple column names are underlined, the primary key is a concatenation of those columns.
- Column description
- Column type; see data type description following
- A list of valid values for the column, as applicable; Appendix A contains a translation of these codes
- One or more reference IDs that provide cross-references to data item descriptions in Sections 2 and 3, as applicable. Columns without reference IDs are generally internal identifiers that link rows in different tables and establish the relational database.

The data types for columns are CHAR, NUMBER, and DATE. A CHAR column can contain a sequence of alphanumeric characters. The number in parentheses is the maximum length of the field. A NUMBER column can contain only the numerals 0 through 9 and the signs + and −. The first number in the parentheses identifies the width of the numeric field. The second number (after the comma) identifies the number of places after the decimal point. A zero indicates that column entries must be integers. A DATE column can contain only a date, formatted as DD-MMM-YY. Reference 4 presents a more detailed description of ORACLE datatypes.

Table 4-2 is intended for users, such as maintenance programmers, who need to know more of the technical specifications for all 64 base tables and 47 views on the VAX, and 14 base tables and 4 views on the PC. Provided for each field are name; data type; length (the number of decimal places is specified if the field is numeric); an indication of whether it is the primary key or part of the primary key; a specification of whether it can contain null values; and whether it is indexed. Fields that are identified as being indexed are those to be used frequently in join operations, in comparison, or in specifying search conditions. Unique indices exist for all fields or concatenations of fields that must have unique values within a particular table row. The last column in the table is for the view entries. It specifies the underlying table from which a particular column within a view is derived.

Table 4-1. SEL Database Tables and Views (1 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
CHANGE		TABLE CONTAINING CRF INFORMATION FOR ALL CHANGES			
	CHANGE_NO	FORM NUMBER OF CRF	CHAR (6)		P63, D82
	PROG_ID	ID UNIQUELY IDENTIFYING EACH PROGRAMMER (FROM TABLE PERSONNEL)	NUMBER (5, 0)		
	SUB_DATE	SUBMISSION DATE OF CRF	DATE		P65, D2
	EFF_ONE	YES/NO FLAG TO INDICATE WHETHER CHANGE WAS MADE TO ONE AND ONLY ONE COMPONENT	CHAR (1)	Y, N	P68, D68
	EFF_ADA	YES/NO FLAG TO INDICATE WHETHER USE OF ADA CONTRIBUTED TO THIS CHANGE	CHAR (1)	Y, N	P69, D76
	EFF_ISO_CH	PROGRAMMER'S EFFORT TO ISOLATE CHANGE	CHAR (10)	1HR, 1DAY, 3DAY, NDAY, NOTDET	P66, D65
	EFF_COM_CH	PROGRAMMER'S EFFORT TO IMPLEMENT CHANGE	CHAR (10)	1HR, 1DAY, 3DAY, NDAY, NOTDET	P67, D66
	EFF_PARPA	YES/NO FLAG TO INDICATE WHETHER PROGRAMMER HAD TO BE AWARE OF PARAMETERS PASSED	CHAR (1)	Y, N	P71, D70
	EFF_OTHER	YES/NO FLAG TO INDICATE WHETHER PROGRAMMER LOOKED AT ANY OTHER COMPONENTS	CHAR (1)	Y, N	P70, D69
	DATE_DETER	DATE ON WHICH NEED FOR CHANGE WAS DETERMINED	DATE		P72, D63
	DATE_COMP	DATE ON WHICH CHANGE WAS COMPLETED	DATE		P73, D64
	NUM_COM_CH	TOTAL NUMBER OF COMPONENTS CHANGED	NUMBER (3, 0)		P74
	NUM_COM_EX	TOTAL NUMBER OF COMPONENTS EXAMINED	NUMBER (2, 0)		P75

Table 4-1. SEL Database Tables and Views (2 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
CHANGE (CONT'D)	CH_TYPE	TYPE OF CHANGE	CHAR (10)	ERRCO, PLANE, IMPRE, IMPCM, IMPUS, IN/DE, OPTSA, ADENC, OTHCH	P76, D67
	FORM_TYPE	TYPE OF DATA COLLECTION FORM	CHAR (6)	CRF	
	STATUS	STATUS OF CRF	CHAR (10)	UNCHK, HCCORRECT, HCERROR, VERAP, CLOSED	
CHANGE_ COM		TABLE CONTAINING CHANGED COMPONENTS ASSOCIATED WITH PARTICULAR CRFs			
	CHANGE_NO	FORM NUMBER OF CRF FROM TABLE CHANGE	CHAR (6)		P63, D82
	COM_NO	ID OF CHANGED COMPONENT FROM TABLE SUB_COM	NUMBER (7, 0)		
CH_ ADAFEAT		TABLE CONTAINING ADA FEATURES THAT WERE INVOLVED IN OR CONTRIBUTED TO PARTICULAR CHANGES			
	CHANGE_NO	FORM NUMBER OF CRF FROM TABLE CHANGE	CHAR (6)		P63, D82
	ADA_FEATURE	FEATURES(S) INVOLVED IN CHANGE IF ADA IS USED AS DESIGN AND IMPLEMENTATION LANGUAGE	CHAR (10)	DATATYPE, SUBPROG, EXCEPT, GEN, PACK, TASK, SYSDEPF, OTHER	P85, D77
CH_ERR_ ARES		TABLE CONTAINING RESOURCES USED IN CORRECTING ERRORS FOR PARTICULAR CHANGES INVOLVING ADA			
	CHANGE_NO	FORM NUMBER OF CRF FROM TABLE CHANGE	CHAR (6)		P63, D82
	ERR_ARES	RESOURCES USED TO CORRECT ERROR CAUSED BY USE OF ADA	CHAR (10)	NOTE, REFMAN, TEAM, MEMORY, NTEAM, OTHER	P86, D80
CH_ERR_ GEN		TABLE CONTAINING ERROR CHARACTERISTICS FOR PARTICULAR CHANGES IDENTIFIED AS ERROR CORRECTIONS			
	CHANGE_NO	FORM NUMBER OF CRF FROM TABLE CHANGE	CHAR (6)		P63, D82

Table 4-1. SEL Database Tables and Views (3 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
CH_ERR_ GEN (CONT'D)	ERR_SOURCE	SOURCE OF ERROR	CHAR (10)	REQMT, FUNSPEC, DESIGN, CODE, PRECH, NOTDET	P77, D71
	ERR_CLASS	CLASS OF ERROR	CHAR (10)	INIT, LOGIC, INTERI, INTERE, DATAVAL, COMPUTE, NOTDET	P78, D72
	ERR_COMIS	YES/NO FLAG TO INDICATE WHETHER ERROR WAS ONE OF COMMISSION	CHAR (1)	Y, N	P79, D74
	ERR_TYPO	YES/NO FLAG TO INDICATE WHETHER ERROR WAS TYPOGRAPHICAL	CHAR (1)	Y, N	P81, D75
	ERR_OMIS	YES/NO FLAG TO INDICATE WHETHER ERROR WAS ONE OF OMISSION	CHAR (1)	Y, N	P80, D73
	ERR_ADOC	YES/NO FLAG TO INDICATE WHETHER ADA COMPILER DOCUMENTATION OR ADA LANGUAGE REFERENCE MANUAL EXPLAINS INVOLVED FEATURES CLEARLY	CHAR (1)	Y, N	P82, D78
	ERR_ACAUSE	CAUSE OF ERROR INVOLVING ADA	CHAR (10)	INTERACT, INCOF, FEATUREM, FEATUREC	P83, D79
CH_ERR_ TOOLS		TABLE CONTAINING TOOLS USED IN CORRECTING ERRORS FOR PARTICULAR CHANGES INVOLVING ADA			
	CHANGE_NO	FORM NUMBER OF CRF FROM TABLE CHANGE	CHAR (6)		P63, D82
	ERR_TOOLS	ADA TOOLS USED THAT AIDED IN DETECTION OR CORRECTION OF ERROR	CHAR (10)	COMPI, SYMDEB, LSE, CMS, SCA, PCA DECTM, OTHER	P87, D81
COMPUTER		TABLE CONTAINING INFORMATION ABOUT COMPUTERS USED ON VARIOUS PROJECTS			
	CPU_NAME	SHORT, UNIQUE NAME IDENTIFYING A PARTICULAR COMPUTER	CHAR (10)		P44, P134, M4, D38
	C_FULL_NAME	COMPUTER FULL NAME	CHAR (20)		M5

Table 4-1. SEL Database Tables and Views (4 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
COM_ PURPOSE		TABLE CONTAINING PURPOSES REPORTED ON COFs FOR EXECUTABLE COMPONENTS			
	COM_NO	ID UNIQUELY IDENTIFYING EACH COMPONENT (FROM TABLE SUB_COM)	NUMBER (7, 0)		
	PURPOSE	MAJOR PURPOSE(S) OF COMPONENT	CHAR (10)	IOPRO, ALCOMP, DATRA, LODEC, CNTRMOD, INTOP, ADAPR, ADADA	P59, D58
COM_ SOURCE		TABLE CONTAINING COF INFORMATION FOR ALL COMPONENTS			
	COM_NO	ID UNIQUELY IDENTIFYING EACH COMPONENT (FROM TABLE SUB_COM)	NUMBER (7,0)		
	PROG_ID	ID UNIQUELY IDENTIFYING EACH PROGRAMMER (FROM TABLE PERSONNEL)	NUMBER (5, 0)		
	FORM_NO	FORM NUMBER OF COF	CHAR (6)		D59
	FORM_TYPE	TYPE OF DATA COLLECTION FORM	CHAR (6)	COF	
	STATUS	STATUS OF COF	CHAR (10)	UNCHK, HCCORRECT, HCERROR, VERAP, CLOSED	
	CREATE_DATE	DATE ON WHICH COMPONENT WAS ENTERED INTO CONTROLLED LIBRARY	DATE		P53, D54
	ORI_TYPE	ORIGIN OF COMPONENT	CHAR (10)	NEW, EXTMO, SLMOD, OLDUC	P56, D56
	COM_TYPE	TYPE OF COMPONENT	CHAR (10)	INCL, JCL, ALC, FORTRAN, PASCAL, NAMELT, DISPLAY, MENDEF, REFDATA, BLOCKDA, ADASUBS, ADASUBB, ADAPACKS, ADAPACKB, ADATASKS, ADATASKB, ADAGENS, ADAGENB, ADAUNSPEC, OTHER	P58, D57
	DIFFICULTY	DEGREE OF DIFFICULTY IN CREATING PARTICULAR COMPONENT	NUMBER (2, 0)	1 TO 5	P57, D55
	SUB_DATE	SUBMISSION DATE OF COF	DATE		P54, D2

Table 4-1. SEL Database Tables and Views (5 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
COM_STAT		TABLE CONTAINING STATISTICS FOR ALL COMPONENTS			
	COM_NO	ID UNIQUELY IDENTIFYING EACH COMPONENT (FROM TABLE SUB_COM)	NUMBER (7, 0)		
	C_EXE_S	NUMBER OF EXECUTABLE STATEMENTS IN COMPONENT	NUMBER (6, 0)		P154
	C_LINE	NUMBER OF SOURCE LINES OF CODE (WITH COMMENTS) IN COMPONENT	NUMBER (6, 0)		P155
	C_C_LINE	NUMBER OF COMMENT LINES IN COMPONENT (NO BLANK LINES)	NUMBER (6, 0)		P156
	C_STMT	NUMBER OF STATEMENTS IN THE COMPONENT	NUMBER (6, 0)		P221
	FINAL_ORIGIN_CAT	ORIGIN CATEGORY ASSIGNED TO THE COMPONENT FOR COMPUTING FINAL STATISTICS	CHAR (10)	NEW, EXTMO, SLMOD, OLDUC	P222
DSF_MEASURE		TABLE CONTAINING DSF MEASUREMENT DATA			
	D_ID	D_ID FROM TABLE PROJ_DSF	NUMBER (10, 0)		
	STATUS_CODE	TYPE OF DSF DATA	CHAR (10)	DESIGN, CODE, SYSTEST, ACCTEST, DISCREP, QUESTIONS, SPECMOD	
	MEASURE_CODE	TYPE OF DSF MEASURE	CHAR (10)	MODDESIGN, MODCODE, SYSTSTONE, SYSTSTPASS, SYSTSTRUN, ACCTSTONE, ACCTSTPASS, ACCTSTRUN, DISCRES, QUESTANS, SPECMODIMP	P196, P198, P200, P204, P208, P210, P212
	MEASURE_VALUE	VALUE OF DSF MEASURE	NUMBER (5, 0)		P196, D181, P198, D183, P200–P202, D185–D186, P204–P206, D189–D190 P208, D193, P210, D195, P212, D197

Table 4-1. SEL Database Tables and Views (6 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
DSF_ TARGET		TABLE CONTAINING DSF TARGET DATA			
	D_ID	D_ID VALUE FROM TABLE PROJ_DSF	NUMBER (10, 0)		
	STATUS_CODE	TYPE OF DSF DATA	CHAR (10)	DESIGN, CODE, SYSTEST, ACCTEST, DISCREP, QUESTIONS, SPECMOD	
	TARGET_CODE	TYPE OF DSF TARGET	CHAR (10)	TOTDESIGN, TOTCODE, TOTSYSTST, TOTACCTST, TOTDISCREP, QUESTSUB, SPECMODREC	P195, P197, P199, P203, P207, P209, P211
	TARGET_VALUE	VALUE OF DSF TARGET	NUMBER (5, 0)		P195, D180, P197, D182, P199, D184, P203, D188, P207, D192, P209, D194, P211, D196
EFF_ACT		TABLE CONTAINING TECHNICAL AND DIRECT MANAGEMENT ACTIVITY HOURS FROM CLPRFs OR PRFs AND SERVICE PERSONNEL HOURS FROM SPFs FOR ALL PROJECT, PERSONNEL, AND WEEK COMBINATIONS			
	EFF_ID	P_ID VALUE FROM TABLE EFF_PROJ OR PS_ID VALUE FROM TABLE EFF_SUB	NUMBER (10, 0)		
	ACTIVITY	ACTIVITY TO WHICH PERSONNEL ARE CHARGING TIME ON CLPRF, PRF, OR SPF	CHAR (10)	ACCTEST, CLACCTEST, CLCREDES, CLINDTEST, CLOTHER, CLPREDES, CLPRETEST, CLRDREVCOD, CLRESPSFR, CLVEREVDES, CLWRCODE, CREDES, DEBUG, INTTEST, OTHER, PREDES, RDREVCOD, RDREVDES, SUPPORT, TSTCODUN, WRCODE	P25–P34, P39–P40, P42–P43, P157–P166

Table 4-1. SEL Database Tables and Views (7 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
EFF_ACT (CONT'D)	ACT_HR	ACTUAL HOURS SPENT IN PARTICULAR ACTIVITY	NUMBER (10, 2)		P25–P34, D23–D32, P39–P40, D44–D45, P42–P43, D47–D48, P157–P166, D199–D208
EFF_FORM		TABLE CONTAINING FORM IDENTIFICATION AND STATUS INFORMATION FOR EACH PROJECT, PROGRAMMER AND WEEK COMBINATION; ENTERED FROM CLPRFs, PRFs, OR SPFs			
	P_ID	P_ID VALUE FROM TABLE EFF_PROJ	NUMBER (10, 0)		
	FORM_NO	FORM NUMBER OF CLPRF, PRF, OR SPF	CHAR (6)		D37, D49, D210
	FORM_TYPE	TYPE OF DATA COLLECTION FORM	CHAR (6)	CLPRF, PRF, SPF	
	STATUS	STATUS OF CLPRF, PRF, OR SPF	CHAR (10)	UNCHK, HCCORRECT, HCERROR, VERAP, CLOSED	
EFF_PROJ		TABLE ASSOCIATING GIVEN PROJECT, PROGRAMMER, AND WEEK COMBINATION WITH SURROGATE KEY (P_ID) FOR USE IN OTHER TABLES			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF CLPRF, PRF, OR SPF	DATE		P23, D22
	PROG_ID	ID UNIQUELY IDENTIFYING EACH PROGRAMMER (FROM TABLE PERSONNEL)	NUMBER (5, 0)		
	P_ID	SURROGATE KEY ASSIGNED TO REPRESENT UNIQUE PROJ_NO, PROG_ID, AND SUB_DATE COMBINATION	NUMBER (10, 0)		

Table 4-1. SEL Database Tables and Views (8 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
EFF_SUB		TABLE ASSOCIATING P_ID AND SUBSYSTEM PREFIX WITH SURROGATE KEY (PS_ID) FOR USE IN OTHER TABLES			
	P_ID	P_ID VALUE FROM TABLE EFF_PROJ	NUMBER (10, 0)		
	SUB_PRE	SUBSYSTEM PREFIX FROM TABLE PROJ_SUB	CHAR (5)		P47, D52, D152
	PS_ID	SURROGATE KEY ASSIGNED TO REPRESENT UNIQUE P_ID AND SUB_PRE COMBINATION	NUMBER (10, 0)		
MAINT_ACT_HRS		TABLE CONTAINING PROGRAMMER MAINTENANCE HOURS FROM WMEFs GROUPED BY ACTIVITIES			
	MAINT_ID	MAINT_ID VALUE FROM TABLE MAINT_PROJ	NUMBER (10, 0)		
	MAINT_ACT	ACTIVITY TO WHICH PROGRAMMER IS CHARGING TIME ON WMEF	CHAR (10)	ISOLATION, REDESIGN, IMPLEMENT, UNSYSTEST, ACCBENTEST, OTHER	P172–P177
	ACT_HR	ACTUAL HOURS SPENT IN PARTICULAR ACTIVITY	NUMBER (10, 2)		P172–P177, D155–D160
MAINT_CHANGE		TABLE CONTAINING INFORMATION FOR ALL MAINTENANCE CHANGES			
	MAINT_CH_NO	FORM NUMBER OF MCRF	CHAR (6)		D178
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	PROG_ID	ID UNIQUELY IDENTIFYING EACH PROGRAMMER (FROM TABLE PERSONNEL)	NUMBER (5, 0)		
	SUB_DATE	SUBMISSION DATE OF MCRF	DATE		P65, D2
	OSMR_NO	OSMR NUMBER	NUMBER (4, 0)		P178, D162
	STATUS	STATUS OF MCRF	CHAR (10)	UNCHK, HCCORRECT, HCERROR, VERAP, CLOSED	

Table 4-1. SEL Database Tables and Views (9 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
MAINT_ CHANGE (CONT'D)	FORM_TYPE	TYPE OF DATA COLLEC-TION FORM	CHAR (6)	MCRF	
	MAINT_CH_ TYPE	TYPE OF MODIFICATION	CHAR (10)	CORRECTION, ENHANCEMNT, ADAPTATION	P179, D163
	CH_CAUSE	CAUSE OF CHANGE	CHAR (10)	REQMTSPEC, DESIGN, CODE, PRECH, OTHER	P180, D164
	MAINT_ISO_CH	PROGRAMMER'S EF-FORT TO ISOLATE CHANGE	CHAR (10)	1HR, 1DAY, 1WEEK, 1MONTH, 1MONTHMORE	P181, D165
	MAINT_COM_ CH	PROGRAMMER'S EF-FORT TO IMPLEMENT CHANGE	CHAR (10)	1HR, 1DAY, 1WEEK, 1MONTH, 1MONTHMORE	P182, D166
	CH_CLASS	CLASS OF CHANGE	CHAR (10)	INIT, LOGIC, INTERI, INTERE, DATAVAL, COMPUTE, OTHER	P184, D168
	EST_LOC_ADD	ESTIMATED NUMBER OF LINES OF CODE ADDED	NUMBER (6, 0)		P185, D169
	EST_LOC_CH	ESTIMATED NUMBER OF LINES OF CODE CHANGED	NUMBER (6, 0)		P186, D170
	EST_LOC_DEL	ESTIMATED NUMBER OF LINES OF CODE DE-LETED	NUMBER (6, 0)		P187, D171
	COMP_ADD	NUMBER OF COM-PONENTS ADDED	NUMBER (4, 0)		P188, D172
	COMP_CH	NUMBER OF COM-PONENTS CHANGED	NUMBER (4, 0)		P189, D173
	COMP_DEL	NUMBER OF COM-PONENTS DELETED	NUMBER (4, 0)		P190, D174
	COMP_ADD_ NEW	NUMBER OF THE ADDED COMPONENTS THAT ARE TOTALLY NEW	NUMBER (4, 0)		P191, D175
	COMP_ADD_ REUSE	NUMBER OF THE ADDED COMPONENTS THAT ARE TOTALLY REUSED (UNCHANGED)	NUMBER (4, 0)		P192, D176
	COMP_ADD_ REMOD	NUMBER OF THE ADDED COMPONENTS THAT ARE REUSED WITH MODIFICATIONS	NUMBER (4, 0)		P193, D177

Table 4-1. SEL Database Tables and Views (10 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
MAINT_CH_ OBJECTS		TABLE CONTAINING CHANGED OBJECTS ASSOCIATED WITH PARTICULAR MCRFs			
	MAINT_CH_NO	FORM NUMBER OF MCRF FROM TABLE MAINT_CHANGE	CHAR (6)		D178
	CH_OBJECT	CHANGED OBJECT	CHAR (10)	REQMTDOC, DESIGNDOC, CODE, SYSDESC, USERGUIDE, OTHER	P183, D167
MAINT_ CLASS_HRS		TABLE CONTAINING PROGRAMMER MAINTENANCE HOURS FROM WMEFs GROUPED BY CLASS OF MAINTENANCE			
	MAINT_ID	MAINT_ID VALUE FROM TABLE MAINT_PROJ	NUMBER (10, 0)		
	MAINT_CLASS	CLASS OF MAINTENANCE TO WHICH PROGRAMMER IS CHARGING TIME ON WMEF	CHAR (10)	CORRECTION, ENHANCEMNT, ADAPTATION, OTHER	P168–P171
	CLASS_HR	ACTUAL HOURS SPENT IN PARTICULAR CLASS OF MAINTENANCE	NUMBER (10, 2)		P168–P171, D151–D154
MAINT_ PROJ		TABLE CONTAINING WMEF DATA. A GIVEN PROJECT, PROGRAMMER, AND WEEK ARE ASSOCIATED WITH SURROGATE KEY (MAINT_ID) FOR USE IN OTHER TABLES			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF WMEF	DATE		P23, D22
	PROG_ID	ID UNIQUELY IDENTIFYING EACH PROGRAMMER (FROM TABLE PERSONNEL)	NUMBER (5, 0)		P24, D21
	MAINT_ID	SURROGATE KEY ASSIGNED TO REPRESENT UNIQUE PROJ_NO, SUB_DATE, AND PROG_ID COMBINATION	NUMBER (10, 0)		
	FORM_NO	FORM NUMBER OF WMEF	CHAR (6)		D161
	FORM_TYPE	TYPE OF DATA COLLECTION FORM	CHAR (6)	WMEF	

Table 4-1. SEL Database Tables and Views (11 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
MAINT_ PROJ (CONT'D)	STATUS	STATUS OF WMEF	CHAR (10)	UNCHK, HCCORRECT, HCERROR, VERAP, CLOSED	
PERSONNEL		TABLE CONTAINING INFORMATION ABOUT PERSONNEL FOR WHOM DATA ARE RECORDED IN THE DATABASE			
	PROG_ID	ID ASSIGNED FOR UNIQUELY IDENTIFYING EACH PERSON SUBMITTING FORMS	NUMBER (5, 0)		
	FORM_NAME	ABBREVIATED NAME AS IT APPEARS ON VARIOUS FORMS	CHAR (15)	THIS FIELD ALSO INCLUDES THE FOLLOWING SERVICES PERSONNEL NAMES: LIBARIAN–LIBRARIANS OTHSUPP–OTHER SUPPORT PERSONNEL PROGMGMT–PROGRAM MANAGEMENT PERSONNEL SECRTARY–SECRETARIES TECHPUBS–TECHNICAL PUBLICATIONS PERSONNEL	P24, M1, D21
	FULL_NAME	FULL DESCRIPTIVE NAME OF PERSON	CHAR (30)		M2
	DATE_ENTRY	DATE ON WHICH PERSONNEL DATA WERE ENTERED INTO DATABASE	DATE		M3
PROJECT		TABLE CONTAINING INFORMATION ABOUT ALL PROJECTS IN THE DATABASE			
	PROJ_NAME	PROJECT NAME	CHAR (8)		P1, D1
	PROJ_NO	ID ASSIGNED FOR UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (3, 0)		
	PROJ_TYPE	PROJECT CATEGORY	CHAR (10)	AGSS, ATTITUDE, DATABASE, GRAPH/UI, MP&A, ORBIT, OTHER REALTIME, SIMULATOR, TOOL	P2, D163

Table 4-1. SEL Database Tables and Views (12 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
PROJECT (CONT'D)	ACTIVE_ STATUS	CURRENT STATUS OF PROJECT	CHAR (10)	ACT_DEV, ACT_MAINT, INACTIVE, DISCONT	P3
PROJ_CPU_ STAT		TABLE CONTAINING AT-COMPLETION COMPUTER RESOURCE STATISTICS FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF PCSF	DATE		P124, D2
	CPU_NAME	SHORT NAME IDENTIFYING COMPUTER USED ON PROJECT (FROM TABLE COMPUTER)	CHAR (10)		P134, M4, D38
	TOTAL_HRS	TOTAL COMPUTER HOURS USED ON PARTICULAR COMPUTER FOR PROJECT	NUMBER (10, 2)		P135, D94
	T_RUN	TOTAL NUMBER OF RUNS ON PARTICULAR COMPUTER FOR PROJECT	NUMBER (6, 0)		P136, D95
PROJ_DSF		TABLE CONTAINING FORM IDENTIFICATION AND STATUS INFORMATION FOR EACH PROJECT, PROGRAMMER, AND WEEK COMBINATION; ENTERED FROM DSFs			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF DSF	DATE		P23, D22
	PROG_ID	ID UNIQUELY IDENTIFYING EACH PROGRAMMER (FROM TABLE PERSONNEL)	NUMBER (5, 0)		
	FORM_NO	FORM NUMBER OF DSF	CHAR (6)		D198
	STATUS	STATUS OF DSF	CHAR (10)	UNCHK, HCCORRECT, HCERROR, VERAP, CLOSED	
	FORM_TYPE	TYPE OF DATA COLLECTION FORM	CHAR (6)	DSF	

Table 4-1. SEL Database Tables and Views (13 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
PROJ_DSF (CONT'D)	D_ID	SURROGATE KEY ASSIGNED TO REPRESENT UNIQUE PROJ_NO, SUB_DATE COMBINATION	NUMBER (10, 0)		
PROJ_EST		TABLE CONTAINING ESTIMATED STATISTICS FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF PEF	DATE		P13, D2
	T_SYS	ESTIMATED TOTAL NUMBER OF SUBSYSTEMS	NUMBER (4, 0)		P14, D14
	T_COM	ESTIMATED TOTAL NUMBER OF COMPONENTS	NUMBER (4, 0)		P15, D15
	T_LINE	ESTIMATED TOTAL SLOC	NUMBER (7, 0)		P16, D16
	T_NEW_LINE	ESTIMATED TOTAL SLOC FOR ALL NEW COMPONENTS	NUMBER (7, 0)		P19, D17
	T_MOD_LINE	ESTIMATED TOTAL SLOC FOR ALL MODIFIED COMPONENTS	NUMBER (7, 0)		P18, D18
	T_OLD_LINE	ESTIMATED TOTAL SLOC FOR ALL REUSED COMPONENTS	NUMBER (7, 0)		P17, D19
	PRO_HR	ESTIMATED TOTAL PROGRAMMER HOURS	NUMBER (10, 2)		P20, D11
	MAN_HR	ESTIMATED TOTAL MANAGEMENT HOURS	NUMBER (10, 2)		P21, D12
	SER_HR	ESTIMATED TOTAL SERVICES HOURS	NUMBER (10, 2)		P22, D13
PROJ_EST_PHASE		TABLE CONTAINING ESTIMATED AND AT-COMPLETION PHASE DATES FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF PCSF OR PEF	DATE		P5, P13, P124, D2
	PHASE_CO	PHASE CODE IDENTIFYING DIFFERENT PHASES IN LIFE OF PROJECT	CHAR (10)	REQNT, DESGN, CODET, SYSTE, ACCTE, CLEAN, MAINT	P6–P21, P125–P131

Table 4-1. SEL Database Tables and Views (14 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
PROJ_EST_ PHASE (CONT'D)	START_DATE	START DATE OF A PARTICULAR PHASE	DATE		P6–P11, D3–D8, P125–P131, D84–D90
	END_DATE	END DATE OF A PARTICULAR PHASE	DATE		P6–P11, D4–D8, D10, P125–P131, D85–D91
PROJ_FORM		TABLE CONTAINING FORM IDENTIFICATION AND STATUS INFORMATION FOR PCSF, PEF, SEF, AND SPF DATA			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF PCSF, PEF, SEF, OR SPF	DATE		P13, P124, D2, P23, D22
	FORM_NO	FORM NUMBER OF PCSF, PEF, SEF, OR SPF	CHAR (6)		D150, D20, D49, D113
	FORM_TYPE	TYPE OF DATA COLLECTION FORM	CHAR (6)	PCSF, PEF, SEF, SPF	
	STATUS	STATUS OF PCSF, PEF, SEF, OR SPF	CHAR (10)	UNCHK, HCCORRECT, HCERROR, VERAP, CLOSED	
PROJ_GRH		TABLE CONTAINING GROWTH HISTORY INFORMATION FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF SPF	DATE		P23, D22
	GR_LINE	TOTAL NUMBER OF LINES OF CODE (WITH COMMENTS) IN PROJECT CONTROLLED SOURCE LIBRARY	NUMBER (7, 0)		P60, D43
	GR_MOD	TOTAL NUMBER OF MODULES IN PROJECT CONTROLLED LIBRARY	NUMBER (4, 0)		P61, D41
	GR_CH	TOTAL NUMBER OF CHANGES RECORDED IN PROJECT CONTROLLED LIBRARY	NUMBER (6, 0)		P62, D42
PROJ_ MESSAGES		TABLE CONTAINING GENERAL PROJECT DESCRIPTION INFORMATION FOR ALL PROJECTS IN DATABASE			

Table 4-1. SEL Database Tables and Views (15 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
PROJ_ MESSAGES (CONT'D)	S_ID	S_ID FROM TABLE PROJ_NOTES	NUMBER (5, 0)		
	LINE_NO	LINE SEQUENCE NUMBER WITHIN A MESSAGE	NUMBER (3, 0)		
	MESSAGES	GENERAL PROJECT DESCRIPTION INFORMATION	CHAR (65)		P4, D62
	SUB_DATE	DATE ON WHICH MESSAGE WAS SUBMITTED	DATE		D2
PROJ_ NOTES		TABLE ASSOCIATING GIVEN PROJECT AND MESSAGE TYPE WITH SURROGATE KEY (S_ID) FOR USE IN THE PROJ_MESSAGES TABLE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	NOTE_TYPE	GENERAL PROJECT DESCRIPTION CODES	CHAR (10)	CLOSEOUT, COMPACCTS, COMPSYS, CONTACTS, CONTRLLIB, DATAAVAIL, FORMSCOL, GENMESS, GHTOOL, LANGUAGES, PROJNAME, TASKNO	P4, D61
	S_ID	SURROGATE KEY ASSIGNED TO REPRESENT UNIQUE PROJ_NO AND NOTE_TYPE COMBINATION	NUMBER (5, 0)		
PROJ_PROD		TABLE CONTAINING WEEKLY COMPUTER RESOURCE USE INFORMATION FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3,0)		
	SUB_DATE	SUBMISSION DATE OF SPF	DATE		P23, D22
	RES_NAME	SHORT NAME IDENTIFYING COMPUTER USED ON A PROJECT (FROM TABLE COMPUTER)	CHAR (10)		P44, M4, D38
	RES_HR	TOTAL CPU HOURS USED IN CURRENT WEEK	NUMBER (10, 2)		P45, D39

Table 4-1. SEL Database Tables and Views (16 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
PROJ_PROD (CONT'D)	RES_RUN	TOTAL RUNS MADE IN CURRENT WEEK	NUMBER (5, 0)		P46, D40
PROJ_SEF		TABLE CONTAINING SUBJECTIVE MEASURES FROM SEFs FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM PROJECT TABLE)	NUMBER (3, 0)		
	MEAS_TYPE	CODES IDENTIFYING SUBJECTIVE PROJECT CHARACTERISTICS	CHAR (10)	PM01, PM02, PM03, PM04, PM05, PM06, ST07, ST08, ST09, ST10, TM11, TM12, TM13, TM14, TM15, PC16, PC17, PC18, PC19, PC20, PC22, PC23, PC24, EN25, EN26, EN27, EN28, EN29, EN30, PT31, PT32, PT33, PT34, PT35, PT36	
	EVALUATE	INTEGER INDICATING THE VALUE OF A PARTICULAR MEAS_TYPE	NUMBER (1, 0)	1 TO 5	P88–P107, D114–D133, P109–P123, D135–D149
PROJ_SEF_SEC		TABLE CONTAINING SECONDARY-LEVEL INFO, AS RECORDED ON SEFs, FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	MEAS_TYPE	CODE IDENTIFYING PROJECT CHARACTERISTICS AND TOOLS USED	CHAR (10)	PC21	
	SECOND_L	SECONDARY LEVEL INFORMATION FOR A PARTICULAR MEAS_TYPE; AT PRESENT, ALL THE CODES STORED HERE ARE FOR "USE OF TOOLS" (PC21)	CHAR (10)	COMPL, LINK, EDIT, GRADIS, REPLP, STRANT, PDLPR, ISPF, SAP, CAT, PANVAL, TESTCO, INTERF, LSE, SYMDEB, CMTOOL, SDE, OTHER	P108, D134
PROJ_STAT		TABLE CONTAINING AT-COMPLETION STATISTICS FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF PCSF	DATE		P124, D2

Table 4-1. SEL Database Tables and Views (17 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
PROJ_STAT (CONT'D)	TECH_MAN_HR	TOTAL TECHNICAL AND MANAGEMENT HOURS USED ON PROJECT	NUMBER (10, 2)		P132, D92
	SER_HR	TOTAL SERVICE HOURS EXPENDED ON PROJECT	NUMBER (10, 2)		P133, D93
	T_SYS	TOTAL NUMBER OF SUBSYSTEMS	NUMBER (4, 0)		P137, D96
	T_COM	TOTAL NUMBER OF COMPONENTS	NUMBER (4, 0)		P138, D97
	T_CH	TOTAL NUMBER OF CHANGES	NUMBER (6, 0)		P139, D98
	T_DOC	TOTAL PAGES OF DOCUMENTATION	NUMBER (6, 0)		P140, D99
	T_LINE	TOTAL SLOC FOR ALL COMPONENTS (INCLUDES BLANK LINES)	NUMBER (7, 0)		P141, D100
	T_NEW_LINE	TOTAL SLOC FOR ALL NEW COMPONENTS	NUMBER (6, 0)		P142, D101
	T_MOD_LINE	TOTAL SLOC FOR ALL SLIGHTLY MODIFIED COMPONENTS	NUMBER (6, 0)		P143, D102
	T_OLD_LINE	TOTAL SLOC FOR ALL REUSED (UNCHANGED) COMPONENTS	NUMBER (6, 0)		P144, D103
	T_COMMENT	TOTAL NUMBER OF COMMENT LINES (BLANK LINES NOT INCLUDED)	NUMBER (6, 0)		P145, D104
	T_EXE_MOD	TOTAL NUMBER OF EXECUTABLE COMPONENTS	NUMBER (4, 0)		P146, D105
	T_NEW_MOD	TOTAL NUMBER OF NEW EXECUTABLE COMPONENTS	NUMBER (4, 0)		P147, D106
	T_MOD_MOD	TOTAL NUMBER OF SLIGHTLY MODIFIED EXECUTABLE COMPONENTS	NUMBER (4, 0)		P148, D107
	T_OLD_MOD	TOTAL NUMBER OF REUSED (UNCHANGED) EXECUTABLE COMPONENTS	NUMBER (4, 0)		P149, D108
	T_EXE_STAT	TOTAL NUMBER OF EXECUTABLE STATEMENTS FOR ALL FORTRAN COMPONENTS	NUMBER (6, 0)		P150, D109

Table 4-1. SEL Database Tables and Views (18 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
PROJ_STAT (CONT'D)	T_NEW_STAT	TOTAL NUMBER OF EXECUTABLE STATEMENTS FOR ALL NEW FORTRAN COMPONENTS	NUMBER (6, 0)		P151, D110
	T_MOD_STAT	TOTAL NUMBER OF EXECUTABLE STATEMENTS FOR ALL SLIGHTLY MODIFIED FORTRAN COMPONENTS	NUMBER (6, 0)		P152, D111
	T_OLD_STAT	TOTAL NUMBER OF EXECUTABLE STATEMENTS FOR ALL REUSED (UNCHANGED) FORTRAN COMPONENTS	NUMBER (6, 0)		P153, D112
	T_STMTS	TOTAL NUMBER OF STATEMENTS	NUMBER (6, 0)		P216,D214
	T_NEW_STMTS	TOTAL NUMBER OF STATEMENTS FOR ALL NEW COMPONENTS	NUMBER (6, 0)		P217, D215
	T_MOD_STMTS	TOTAL NUMBER OF STATEMENTS FOR ALL SLIGHTLY MODIFIED COMPONENTS	NUMBER (6, 0)		P218, D216
	T_OLD_STMTS	TOTAL NUMBER OF STATEMENTS FOR ALL REUSED (UNCHANGED) COMPONENTS	NUMBER (6, 0)		P220, D218
	T_EXTMO_LINE	TOTAL SLOC FOR ALL EXTENSIVELY MODIFIED COMPONENTS	NUMBER (6, 0)		P213, D211
	T_EXTMO_MOD	TOTAL NUMBER OF EXTENSIVELY MODIFIED EXECUTABLE COMPONENTS	NUMBER (4, 0)		P214, D212
	T_EXTMO_STAT	TOTAL NUMBER OF EXECUTABLE STATEMENTS FOR ALL EXTENSIVELY MODIFIED FORTRAN COMPONENTS	NUMBER (6, 0)		P215, D213
	T_EXTMO_ STMTS	TOTAL NUMBER OF STATEMENTS FOR ALL EXTENSIVELY MODIFIED COMPONENTS	NUMBER (6, 0)		P219, D217
PROJ_SUB		TABLE ASSOCIATING PROJECT AND SUBSYSTEM WITH SURROGATE KEY (SUBSY_ID) THAT UNIQUELY IDENTIFIES THE SUBSYSTEM FOR USE IN OTHER TABLES			

Table 4-1. SEL Database Tables and Views (19 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
PROJ_SUB (CONT'D)	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT (FROM TABLE PROJECT)	NUMBER (3, 0)		
	SUB_PRE	SUBSYSTEM PREFIX	CHAR (5)		P47, P84, D50
	SUB_DATE	DATE SUBSYSTEM WAS SUBMITTED	DATE		P50, P2
	SUBSY_ID	SURROGATE KEY ASSIGNED TO REPRESENT UNIQUE PROJ_NO AND SUB_PRE COMBINATION	NUMBER (5, 0)		
SPECIAL_ ACT		TABLE CONTAINING PROGRAMMER ACTIVITY HOURS FROM CLPRFs OR PRFs (PART C) FOR ALL PROJECT, PROGRAMMER, AND WEEK COMBINATIONS			
	EFF_ID	P_ID VALUE FROM TABLE EFF_PROJ OR PS_ID VALUE FROM TABLE EFF_SUB	NUMBER (10, 0)		
	SP_ACTIVITY	SPECIAL ACTIVITY TO WHICH PROGRAMMER IS CHARGING TIME ON CLPRF OR PRF	CHAR (10)	CLMETHOD, DOCUMENT, ENHANCE, REUSE, REWORK	P35–P38, P167
	ACT_HR	ACTUAL HOURS SPENT IN A PARTICULAR ACTIVITY	NUMBER (10, 2)		P35–P38, D33–D36, P167, D209
SUBSYSTEM		TABLE CONTAINING INFORMATION FOR PARTICULAR SUBSYSTEMS, AS RECORDED ON SIFs			
	SUBSY_ID	ID UNIQUELY IDENTIFYING EACH SUBSYSTEM (FROM TABLE PROJ_SUB)	NUMBER (5, 0)		
	NAME	SUBSYSTEM DESCRIPTIVE NAME	CHAR (40)		P48, D51
	FUNCTION	SPECIFIC FUNCTION THE SUBSYSTEM PERFORMS	CHAR (10)	USERINT, DPDC, REALTIME, GRAPH, CPEXEC, SYSSERV, MATHCOMP	P49, D52
SUB_COM		TABLE ASSOCIATING SUBSYSTEM AND COMPONENT NAME WITH SURROGATE KEY THAT UNIQUELY IDENTIFIES THE COMPONENT FOR USE IN OTHER TABLES			

Table 4-1. SEL Database Tables and Views (20 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
SUB_COM (CONT'D)	SUBSY_ID	ID UNIQUELY IDENTIFYING EACH SUBSYSTEM (FROM TABLE PROJ_SUB)	NUMBER (5, 0)		
	COM_NAME	COMPONENT DESCRIPTIVE NAME	CHAR (40)		P51, P84, D53
	COM_NO	SURROGATE KEY ASSIGNED TO REPRESENT UNIQUE SUBSY_ID AND COM_NAME COMBINATION	NUMBER (7, 0)		
	COM_DATE	DATE ON WHICH COMPONENT IS ENTERED INTO DATABASE	DATE		P52, D2
VALIDATION		TABLE THAT IDENTIFIES VALID CODES USED IN VARIOUS FIELDS IN DATABASE AND PROVIDES DESCRIPTIONS FOR THEM			
	F_NAME	FIELD NAME FOR WHICH CODE IS VALID	CHAR (20)	SEE APPENDIX A FOR A DESCRIPTION OF ALL CODES AND VALUES	
	CODE	ABBREVIATED CODE	CHAR (10)		
	VALUE	FULL DESCRIPTION OF CODE	CHAR (75)		
V_CLEANROOM ACT		VIEW CONTAINING PERSONNEL ACTIVITY HOURS FROM CLPRFs (FROM TABLE EFF_ACT) THAT ARE CONVERTED INTO PRF ACTIVITY HOURS			
	EFF_ID	SAME AS EFF_ID IN EFF_ACT	NUMBER (10)		
	ACTIVITY	SAME AS ACTIVITY IN EFF_ACT	CHAR (8)		
	ACT_HR	SAME AS ACT_HR IN EFF_ACT	NUMBER		
V_CLEANROOM_PROJECTS		VIEW THAT JOINS THE PROJECT, PROJ_NOTES, AND PROJ_MESSAGES TABLES			
	PROJ_NAME	SAME AS PROJ_NAME IN PROJECT	CHAR (8)		

Table 4-1. SEL Database Tables and Views (21 of 21)

Table or View Name	Column Name	Description	Type	Valid Code/ Value	Reference ID
V_PROJ_COM		VIEW THAT JOINS THE PROJECT, PROJ_SUB, AND SUB_COM TABLES			
	PROJ_NAME	SAME AS PROJ_NAME IN PROJECT	CHAR (8)		
	SUB_PRE	SAME AS SUB_PRE IN PROJ_SUB	CHAR (5)		
	COM_NAME	SAME AS COM_NAME IN SUB_COM	CHAR (40)		
	COM_NO	SAME AS COM_NO IN SUB_COM	NUMBER (7, 0)		
V_PROJ_SUB_ACT		VIEW THAT JOINS THE PROJECT, EFF_PROJ, EFF_SUB, AND EFF_ACT TABLES			
	PROJ_NAME	SAME AS PROJ_NAME IN PROJECT	CHAR (8)		
	SUB_PRE	SAME AS SUB_PRE IN EFF_SUB	CHAR (5)		
	ACTIVITY	SAME AS ACTIVITY IN EFF_ACT	CHAR (10)		
	ACT_HR	SAME AS ACT_HR IN EFF_ACT	NUMBER (10, 2)		
V_SUBSYSTEM_INFO		VIEW THAT JOINS THE PROJECT, PROJ_SUB, AND SUBSYSTEM TABLES			
	SUB_PRE	SAME AS SUB_PRE IN PROJ_SUB	CHAR (5)		
	NAME	SAME AS NAME IN SUB-SYSTEM	CHAR (40)		
	FUNCTION	SAME AS FUNCTION IN SUBSYSTEM	CHAR (10)		
	SUB_DATE	SAME AS SUB_DATE IN PROJECT	DATE		
	PROJ_NAME	SAME AS PROJ_NAME IN PROJECT	CHAR (8)		

Table 4-2. SEL Database Tables and Views—Technical Specifications (1 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
AUTHORIZE	ORA_USER_ID	CHAR	20	PK	N. NULL		USER_CLASS
	ACCESS_TYPE	CHAR	10		N. NULL		USER_CLASS_ACCESS
CHANGE	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX	
	PROG_ID	NUMBER	5, 0		N. NULL	INDEX	
	SUB_DATE	DATE	9		N. NULL	INDEX	
	EFF_ONE	CHAR	1		NULL		
	EFF_ADA	CHAR	1		NULL		
	EFF_ISO_CH	CHAR	10		NULL		
	EFF_COM_CH	CHAR	10		NULL		
	EFF_PARPA	CHAR	1		NULL		
	EFF_OTHER	CHAR	1		NULL		
	DATE_DETER	DATE	9		NULL		
	DATE_COMP	DATE	9		NULL		
	NUM_COM_CH	NUMBER	3, 0		NULL		
	NUM_COM_EX	NUMBER	2, 0		NULL		
	CH_TYPE	CHAR	10		NULL	INDEX	
	FORM_TYPE	CHAR	6		N. NULL		
	STATUS	CHAR	10		N. NULL	INDEX	

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (2 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
CHANGE_COM	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX	
	COM_NO	NUMBER	7, 0	PK	N. NULL	U. INDEX INDEX	
CH_ADAFEAT	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX	
	ADA_FEATURE	CHAR	10		N. NULL	U. INDEX	
CH_ERR_ARES	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX	
	ERR_ARES	CHAR	10		N. NULL	U. INDEX	
CH_ERR_GEN	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX	
	ERR_SOURCE	CHAR	10		NULL		
	ERR_CLASS	CHAR	10		NULL		
	ERR_COMIS	CHAR	1		NULL		
	ERR_TYPO	CHAR	1		NULL		
	ERR_OMIS	CHAR	1		NULL		
	ERR_ADOC	CHAR	1		NULL		
	ERR_ACAUSE	CHAR	10		NULL	INDEX	
CH_ERR_TOOLS	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX	
	ERR_TOOLS	CHAR	10		N. NULL	U. INDEX	

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (3 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
CLOSE_COF	FORM_NO	CHAR	6	PK	N. NULL		
	ORI_TYPE	CHAR	10		NULL		
	COM_TYPE	CHAR	10		NULL		
	DIFFICULTY	NUMBER	2, 0		NULL		
	SUB_DATE	DATE	9		NULL		
	STATUS	CHAR	10		NULL		
	PURPOSE	CHAR	10		NULL		
	COM_NO	NUMBER	7, 0		NULL		
CLOSE_COM_NO_ ORIGIN	COM_NO	NUMBER	7, 0	PK	NULL		
	SUB_PRE	CHAR	5		NULL		
	COM_NAME	CHAR	40		NULL		
	FINAL_ORIGIN_ STATE	CHAR	10		NULL		
CLOSE_CRF	CHANGE_NO	CHAR	6	PK	N. NULL		
	DATE_DETER	DATE	9		NULL		
	DATE_COMP	DATE	9		NULL		
	CH_TYPE	CHAR	10		NULL		
	EFF_ADA	CHAR	1		NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (4 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
CLOSE_CRF (CONT'D)	ADA_FEATURE	CHAR	10		NULL		
	STATUS	CHAR	10		NULL		
	SUB_DATE	DATE	9		NULL		
	EFF_ISO_CH	CHAR	10		NULL		
	EFF_COM_CH	CHAR	10		NULL		
	NUM_COM_CH	NUMBER	2, 0		NULL		
CLOSE_CRF_ERR	CHANGE_NO	CHAR	6	PK	N. NULL		
	SOURCE	CHAR	10		NULL		
	CLASS	CHAR	10		NULL		
	OMIS	CHAR	1		NULL		
	COMIS	CHAR	1		NULL		
	TYPO	CHAR	1		NULL		
	ADOC	CHAR	1		NULL		
	ACAUSE	CHAR	10		NULL		
	ARES	CHAR	10		NULL		
	TOOLS	CHAR	10		NULL		
COMPUTER	CPU_NAME	CHAR	10	PK	N. NULL	U. INDEX	
	C_FULL_NAME	CHAR	20		N. NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (5 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
COM_PURPOSE	COM_NO	NUMBER	7, 0	PK	N. NULL	U. INDEX	
	PURPOSE	CHAR	10		N. NULL	U. INDEX	
COM_SOURCE	COM_NO	NUMBER	7, 0	PK	N. NULL	U. INDEX	
	PROG_ID	NUMBER	5, 0		NULL		
	FORM_NO	CHAR	6		N. NULL	U. INDEX	
	FORM_TYPE	CHAR	6		N. NULL		
	STATUS	CHAR	10		N. NULL	INDEX	
	CREATE_DATE	DATE	9		NULL	INDEX	
	ORI_TYPE	CHAR	10		NULL		
	COM_TYPE	CHAR	10		NULL		
	DIFFICULTY	NUMBER	2, 0		NULL		
	SUB_DATE	DATE	9		NULL	INDEX	
COM_STAT	COM_NO	NUMBER	7, 0	PK	N. NULL	U. INDEX	
	C_EXE_S	NUMBER	6, 0		NULL		
	C_LINE	NUMBER	6, 0		NULL		
	C_C_LINE	NUMBER	6, 0		NULL		
	C_STMT	NUMBER	6, 0		NULL		
	FINAL_ORIGIN_CAT	CHAR	10		NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (6 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
CRF_TEMP_CHANGE_COM	USER_ID	NUMBER		PK	N. NULL	U. INDEX	
	SUB_PRE	CHAR	5	PK	N. NULL	U. INDEX	
	COM_NAME	CHAR	40	PK	N. NULL	U. INDEX	
	COM_NO	NUMBER	7		N. NULL		
DSF_MEASURE	D_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	STATUS_CODE	CHAR	10	PK	N. NULL	U. INDEX	
	MEASURE_CODE	CHAR	10	PK	N. NULL	U. INDEX	
	MEASURE_VALUE	NUMBER	5		N. NULL		
DSF_TARGET	D_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	STATUS_CODE	CHAR	10	PK	N. NULL	U. INDEX	
	TARGET_CODE	CHAR	10	PK	N. NULL	U. INDEX	
	TARGET_VALUE	NUMBER	5		N. NULL		
DUMMY	HIDDEN	CHAR	1		NULL		
EFF_ACT	EFF_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	ACTIVITY	CHAR	10	PK	N. NULL	U. INDEX	
	ACT_HR	NUMBER	10, 2		N. NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (7 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
EFF_FORM	P_ID	NUMBER	10, 0	PK	N. NULL	INDEX	
	FORM_NO	CHAR	6		N. NULL	INDEX	
	FORM_TYPE	CHAR	6		N. NULL		
	STATUS	CHAR	10		N. NULL	INDEX	
EFF_PROJ	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX	
	PROG_ID	NUMBER	5, 0	PK	N. NULL	U. INDEX	
	P_ID	NUMBER	10, 0		N. NULL	U. INDEX	
EFF_SUB	P_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	SUB_PRE	CHAR	5	PK	N. NULL	U. INDEX	
	PS_ID	NUMBER	10, 0		N. NULL	U. INDEX	
GENERATE_SAT_DAY	SCRIPT_NO	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	SAT_DAY	DATE	9	PK	N. NULL	U. INDEX	
IMP_TABLE_NAME[4]	ORA_USER	CHAR	20	PK	N. NULL	U. INDEX	
	TABLE_NAME	CHAR	40	PK	N. NULL	U. INDEX	
MAINT_ACT_HRS	MAINT_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	MAINT_ACT	CHAR	10	PK	N. NULL	U. INDEX	
	ACT_HR	NUMBER	10, 2		N. NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

[4]EMPTY TABLE; RETAINED FOR FUTURE USE

Table 4-2. SEL Database Tables and Views—Technical Specifications (8 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
MAINT_CHANGE	MAINT_CH_NO	CHAR	6	PK	N. NULL	U. INDEX	
	PROJ_NO	NUMBER	3, 0		N. NULL	INDEX	
	PROG_ID	NUMBER	5, 0		N. NULL		
	SUB_DATE	DATE	9		N. NULL		
	OSMR_NO	NUMBER	4, 0		N. NULL		
	STATUS	CHAR	10		N. NULL		
	FORM_TYPE	CHAR	6		N. NULL		
	MAINT_CH_TYPE	CHAR	10		N. NULL		
	CH_CAUSE	CHAR	10		N. NULL		
	MAINT_ISO_CH	CHAR	10		N. NULL		
	MAINT_COM_CH	CHAR	10		N. NULL		
	CH_CLASS	CHAR	10		N. NULL		
	EST_LOC_ADD	NUMBER	6, 0		NULL		
	EST_LOC_CH	NUMBER	6, 0		NULL		
	EST_LOC_DEL	NUMBER	6, 0		NULL		
	COMP_ADD	NUMBER	4, 0		NULL		
	COMP_CH	NUMBER	4, 0		NULL		
	COMP_DEL	NUMBER	4, 0		NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (9 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
MAINT_CHANGE (CONT'D)	COMP_ADD_NEW	NUMBER	4, 0		NULL		
	COMP_ADD_REUSE	NUMBER	4, 0		NULL		
	COMP_ADD_REMOD	NUMBER	4, 0		NULL		
MAINT_CH_OBJECTS	MAINT_CH_NO	CHAR	6	PK	N. NULL	U. INDEX	
	CH_OBJECT	CHAR	10		N. NULL	U. INDEX	
MAINT_CLASS_HRS	MAINT_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	MAINT_CLASS	CHAR	10	PK	N. NULL	U. INDEX	
	CLASS_HR	NUMBER	10, 2		N. NULL		
MAINT_PROJ	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX	
	PROG_ID	NUMBER	5, 0	PK	N. NULL	U. INDEX	
	MAINT_ID	NUMBER	10, 0		N. NULL	U. INDEX	
	FORM_NO	CHAR	6		N. NULL	INDEX	
	FORM_TYPE	CHAR	6		N. NULL		
	STATUS	CHAR	10		N. NULL		
PC_SEQNO	TABLE_NAME	CHAR	30	PK	N. NULL	U. INDEX	
	FIELD_NAME	CHAR	30	PK	N. NULL	U. INDEX	

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (10 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
PC_SEQNO (CONT'D)	MAX_SEQNO	NUMBER	10, 0		N. NULL		
PERM_SCRIPT	ORA_USER	CHAR	20	PK	N. NULL	U. INDEX	
	SCRIPT_NAME	CHAR	20	PK	N. NULL	U. INDEX	
	SCRIPT_NO	NUMBER	10, 0		N. NULL	U. INDEX	
	OUT_ROUTING	CHAR	20		N. NULL		
	OUT_FILE	CHAR	20		N. NULL		
PERSONNEL	PROG_ID	NUMBER	5, 0	PK	N. NULL	U. INDEX	
	FORM_NAME	CHAR	15		N. NULL	U. INDEX	
	FULL_NAME	CHAR	30		NULL		
	DATE_ENTRY	DATE	9		N. NULL		
PROJECT	PROJ_NAME	CHAR	8	PK	N. NULL	U. INDEX	
	PROJ_NO	NUMBER	3, 0		N. NULL	U. INDEX	
	PROJ_TYPE	CHAR	10		NULL		
	ACTIVE_STATUS	CHAR	10		NULL		
PROJ_CPU_STAT	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX	
	CPU_NAME	CHAR	10	PK	N. NULL	U. INDEX	

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (11 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
PROJ_CPU_STAT (CONT'D)	TOTAL_HRS	NUMBER	10, 2		NULL		
	T_RUN	NUMBER	6, 0		NULL		
PROJ_DSF	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX	
	PROG_ID	NUMBER	5, 0		N. NULL		
	FORM_NO	CHAR	6		N. NULL	U. INDEX	
	STATUS	CHAR	10		N. NULL		
	FORM_TYPE	CHAR	6		N. NULL		
	D_ID	NUMBER	10, 0		N. NULL	U. INDEX	
PROJ_EST	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX	
	T_SYS	NUMBER	4, 0		NULL		
	T_COM	NUMBER	4, 0		NULL		
	T_LINE	NUMBER	7, 0		NULL		
	T_NEW_LINE	NUMBER	7, 0		NULL		
	T_MOD_LINE	NUMBER	7, 0		NULL		
	T_OLD_LINE	NUMBER	7, 0		NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (12 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
PROJ_EST (CONT'D)	PRO_HR	NUMBER	10, 2		NULL		
	MAN_HR	NUMBER	10, 2		NULL		
	SER_HR	NUMBER	10, 2		NULL		
PROJ_EST_PHASE	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX	
	PHASE_CO	CHAR	10	PK	N. NULL	U. INDEX	
	START_DATE	DATE	9		N. NULL		
	END_DATE	DATE	9		NULL		
PROJ_FORM	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX	
	FORM_NO	CHAR	6		N. NULL	U. INDEX	
	FORM_TYPE	CHAR	6	PK	N. NULL	U. INDEX INDEX	
	STATUS	CHAR	10		N. NULL	INDEX	
PROJ_GRH	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX	
	GR_LINE	NUMBER	7, 0		NULL		
	GR_MOD	NUMBER	4, 0		NULL		
	GR_CH	NUMBER	6, 0		NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (13 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
PROJ_MESSAGES	S_ID	NUMBER	5, 0	PK	N. NULL	U. INDEX	
	LINE_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	MESSAGES	CHAR	65		N. NULL		
	SUB_DATE	DATE	9		N. NULL		
PROJ_NOTES	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	NOTE_TYPE	CHAR	10	PK	N. NULL	U. INDEX	
	S_ID	NUMBER	5, 0		N. NULL	U. INDEX	
PROJ_PROD	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX	
	RES_NAME	CHAR	10	PK	N. NULL	U. INDEX	
	RES_HR	NUMBER	10, 2		NULL		
	RES_RUN	NUMBER	5, 0		NULL		
PROJ_SEF	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	MEAS_TYPE	CHAR	10	PK	N. NULL	U. INDEX	
	EVALUATE	NUMBER	1, 0		NULL		
PROJ_SEF_SEC	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	MEAS_TYPE	CHAR	10	PK	N. NULL	U. INDEX	
	SECOND_L	CHAR	10	PK	N. NULL	U. INDEX	

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (14 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
PROJ_STAT	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9		N. NULL		
	TECH_MAN_HR	NUMBER	10, 2		NULL		
	SER_HR	NUMBER	10, 2		NULL		
	T_SYS	NUMBER	4, 0		NULL		
	T_COM	NUMBER	4, 0		NULL		
	T_CH	NUMBER	6, 0		NULL		
	T_DOC	NUMBER	6, 0		NULL		
	T_LINE	NUMBER	7, 0		NULL		
	T_NEW_LINE	NUMBER	6, 0		NULL		
	T_MOD_LINE	NUMBER	6, 0		NULL		
	T_OLD_LINE	NUMBER	6, 0		NULL		
	T_COMMENT	NUMBER	6, 0		NULL		
	T_EXE_MOD	NUMBER	4, 0		NULL		
	T_NEW_MOD	NUMBER	4, 0		NULL		
	T_MOD_MOD	NUMBER	4, 0		NULL		
	T_OLD_MOD	NUMBER	4, 0		NULL		
	T_EXE_STAT	NUMBER	6, 0		NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (15 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
PROJ_STAT (CONT'D)	T_NEW_STAT	NUMBER	6, 0		NULL		
	T_MOD_STAT	NUMBER	6, 0		NULL		
	T_OLD_STAT	NUMBER	6, 0		NULL		
	T_STMTS	NUMBER	6, 0		NULL		
	T_NEW_STMTS	NUMBER	6, 0		NULL		
	T_MOD_STMTS	NUMBER	6, 0		NULL		
	T_OLD_STMTS	NUMBER	6, 0		NULL		
	T_EXTMO_LINE	NUMBER	6, 0		NULL		
	T_EXTMO_MOD	NUMBER	4, 0		NULL		
	T_EXTMO_STAT	NUMBER	6, 0		NULL		
	T_EXTMO_STMTS	NUMBER	6, 0		NULL		
PROJ_SUB	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_PRE	CHAR	5	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9		N. NULL		
	SUBSY_ID	NUMBER	5, 0		N. NULL	U. INDEX	
REP_CODES	CODE	CHAR	10	PK	N. NULL	U. INDEX	
	VALUE	CHAR	30		N. NULL		
	FUNCTION	CHAR	15		N. NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (16 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
REP_CONDITIONS[4]	SCRIPT_NO	NUMBER	10, 0	PK	N. NULL		
	REPORT_SEQ	NUMBER	3, 0	PK	N. NULL		
	PROJ_TYPE	CHAR	10		NULL		
	NUM_COM	NUMBER	5, 0		NULL		
	LINES_OF_CODE	NUMBER	5, 0		NULL		
	START_DATE	DATE	9		NULL		
	END_DATE	DATE	9		NULL		
SCRIPT_PROJECTS	SCRIPT_NO	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	REPORT_SEQ	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	PROJ_NAME	CHAR	8	PK	N. NULL	U. INDEX	
SCRIPT_REPORT	SCRIPT_NO	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	REPORT_SEQ	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	REPORT_CODE	CHAR	10		NULL		
	REPORT_TYPE	CHAR	20		N. NULL		
	REPORT_TYPE_ SELECTION	CHAR	10		NULL		
SEQNO	TABLE_NAME	CHAR	30	PK	N. NULL	U. INDEX	
	FIELD_NAME	CHAR	30	PK	N. NULL	U. INDEX	

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

[4]EMPTY TABLE; RETAINED FOR FUTURE USE

Table 4-2. SEL Database Tables and Views—Technical Specifications (17 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
SEQNO (CONT'D)	MAXSEQNO	NUMBER	10, 0		N. NULL		
SPECIAL_ACT	EFF_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	SP_ACTIVITY	CHAR	10	PK	N. NULL	U. INDEX	
	ACT_HR	NUMBER	10, 2		N. NULL		
SUBSYSTEM	SUBSY_ID	NUMBER	5, 0	PK	N. NULL	U. INDEX	
	NAME	CHAR	40		N. NULL		
	FUNCTION	CHAR	10		NULL		
SUB_COM	SUBSY_ID	NUMBER	5, 0	PK	N. NULL	U. INDEX	
	COM_NAME	CHAR	40	PK	N. NULL	U. INDEX	
	COM_NO	NUMBER	7, 0		N. NULL	U. INDEX	
	COM_DATE	DATE	9		N. NULL		
TABLE_PRIVILEGE	TABLE_NAME	CHAR	40	PK	N. NULL	U. INDEX	
	USER_CLASS	CHAR	20	PK	N. NULL	U. INDEX	
	SELECT_PRIV	CHAR	1		NULL		
	INSERT_PRIV	CHAR	1		NULL		
	UPDATE_PRIV	CHAR	1		NULL		
	DELETE_PRIV	CHAR	1		NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (18 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
TABLE_PRIVILEGE (CONT'D)	ALTER_PRIV	CHAR	1		NULL		
	INDEX_PRIV	CHAR	1		NULL		
TEMP_ACTIVITY	ACTIVITY	CHAR	10		N. NULL		
	SAT_DAY	DATE	9		NULL		
	HOURS	NUMBER	10, 2		NULL		
	PROJ_NO	NUMBER	3, 0	PK	N. NULL		
	SUB_HR	NUMBER	10, 2		NULL		
	FLAG	CHAR	4		NULL		
	SCRIPT_NO	NUMBER	10	PK	NULL		
TEMP_FORMCT	SUB_DATE	DATE	9		NULL		
	PROG_ID	NUMBER	5, 0		NULL		
	FORM_TYPE	CHAR	6		NULL		
	PROJ_NO	NUMBER	3, 0	PK	NULL		
	SCRIPT_NO	NUMBER	10, 0	PK	NULL		
TEMP_MANHRS	FORM_NAME	CHAR	15		N. NULL		
	SAT_DAY	DATE	9		NULL		
	HOURS	NUMBER	10, 2		NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (19 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
TEMP_MANHRS (CONT'D)	PROJ_NO	NUMBER	3, 0	PK	N. NULL		
	PROG_ID	NUMBER	5, 0		NULL		
	SUB_HR	NUMBER	10, 2		NULL		
	FLAG	CHAR	4		NULL		
	P_ID	NUMBER	10, 0		NULL		
	SCRIPT_NO	NUMBER	10, 0	PK	NULL		
TEMP_SCRIPT	SCRIPT_NO	NUMBER	10, 0	PK	N. NULL	U. INDEX	
	ORA_USER	CHAR	20		N. NULL		
	PROCESS_ID	CHAR	20		N. NULL		
	OUT_ROUTING	CHAR	20		N. NULL		
	OUT_FILE	CHAR	20		NULL		
	RUN_STATUS	CHAR	10		N. NULL		
	DELETE_STATUS	CHAR	10		N. NULL		
TEMP_SERVHRS	FORM_NAME	CHAR	15		N. NULL		
	SAT_DAY	DATE	9		NULL		
	HOURS	NUMBER	10, 2		NULL		
	PROJ_NO	NUMBER	3, 0	PK	N. NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (20 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
TEMP_SERVHRS (CONT'D)	PROG_ID	NUMBER	5, 0		NULL		
	FLAG	CHAR	4		NULL		
	P_ID	NUMBER	10, 0		NULL		
	SCRIPT_NO	NUMBER	10, 0	PK	NULL		
T_COM_STAT	COM_NO	NUMBER	7, 0	PK	N. NULL		
	C_EXE_S	NUMBER	6, 0		NULL		
	C_LINE	NUMBER	6, 0		NULL		
	C_C_LINE	NUMBER	6, 0		NULL		
	C_STMT	NUMBER	6, 0		NULL		
	FINAL_ORIGIN_CAT	CHAR	10		NULL		
USER_CLASS	ORA_USER_ID	CHAR	20	PK	N. NULL	U. INDEX	
	USER_CLASS	CHAR	20		N. NULL		
USER_CLASS_ ACCESS	USER_CLASS	CHAR	20	PK	N. NULL	U. INDEX	
	ACCESS_TYPE	CHAR	10	PK	N. NULL	U. INDEX	
VALIDATION	F_NAME	CHAR	20	PK	N. NULL	U. INDEX	
	CODE	CHAR	10	PK	N. NULL	U. INDEX	
	VALUE	CHAR	75		N. NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (21 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
VAL_ACTIVE_STATUS	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_ACTIVITY	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_ADA_FEATURE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_CH_CAUSE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_CH_CLASS	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_CH_OBJECT	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_CH_TYPE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_CL_ACTIVITY	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_COM_CH	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (22 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
VAL_COM_PURPOSE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_COM_TYPE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_DATA_AVAIL[5]	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_DSF_MEASURE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_DSF_STATUS	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_DSF_TARGET	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_ERR_ACAUSE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_ERR_ARES	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_ERR_CLASS	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

[5]VIEW DOES NOT EXIST; DEFINITION RETAINED FOR FUTURE USE

Table 4-2. SEL Database Tables and Views—Technical Specifications (23 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
VAL_ERR_SOURCE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_ERR_TOOLS	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_FINAL_ORIGIN_CAT	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_ISO_CH	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_MAINT_ACT	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_MAINT_CH_TYPE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_MAINT_CLASS	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_MAINT_COM_CH	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (24 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
VAL_MAINT_ISO_CH	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_MEAS_TYPE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_NOTE_TYPE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_ORI_TYPE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_PHASE_CO	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_PROJ_TYPE	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_QA_STATUS	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_REPORT_CODE[6]	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_SECOND_L	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

[6]CORRESPONDING VALIDATION CODES NOT DEFINED; VIEW RETAINED FOR FUTURE USE

Table 4-2. SEL Database Tables and Views—Technical Specifications (25 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
VAL_SP_ACTIVITY	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_STATUS	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_S_FUNCTION	CODE	CHAR	10	PK	N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
V_CLEANROOM_ACT	EFF_ID	NUMBER	10, 0	PK	N. NULL		EFF_ACT
	ACTIVITY	CHAR	8		N. NULL		EFF_ACT
	ACT_HR	NUMBER			N. NULL		EFF_ACT
V_CLEANROOM_ PROJECTS	PROJ_NAME	CHAR	8		N. NULL		PROJECT
V_PERM_SCRIPT	SCRIPT_NAME	CHAR	20		N. NULL		PERM_SCRIPT
V_PROJ_COM	PROJ_NAME	CHAR	8	PK	N. NULL		PROJECT
	SUB_PRE	CHAR	5	PK	N. NULL		PROJ_SUB
	COM_NAME	CHAR	40	PK	N. NULL		SUB_COM
	COM_NO	NUMBER	7, 0		N. NULL		SUB_COM
V_PROJ_SUB_ACT	PROJ_NAME	CHAR	8		N. NULL		PROJECT
	SUB_PRE	CHAR	5		N. NULL		EFF_SUB

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (26 of 30)

VAX Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
V_PROJ_SUB_ACT (CONT'D)	ACTIVITY	CHAR	10		N. NULL		EFF_ACT
	ACT_HR	NUMBER	10, 2		N. NULL		EFF_ACT
V_REP_CODES_ CRITERIA	VALUE	CHAR	30		N. NULL		REP_CODES
V_SEQNO	TABLE_NAME	CHAR	30		N. NULL		SEQNO
	FIELD_NAME	CHAR	30		N. NULL		SEQNO
	MAXSEQNO	NUMBER	10, 0		N. NULL		SEQNO
V_SUBSYSTEM_INFO	SUB_PRE	CHAR	5	PK	N. NULL		PROJ_SUB
	NAME	CHAR	40		N. NULL		SUBSYSTEM
	FUNCTION	CHAR	10		NULL		SUBSYSTEM
	SUB_DATE	DATE	9		N. NULL		PROJ_SUB
	PROJ_NAME	CHAR	8	PK	N. NULL		PROJECT

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (27 of 30)

PC Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
AUTHORIZE	ORA_USER_ID	CHAR	20		N. NULL		USER_CLASS
	ACCESS_TYPE	CHAR	10		N. NULL		USER_CLASS_ACCESS
DSFPLOTS	PROJ_NAME	CHAR	8		N. NULL		
DSF_CLOSEOUT	PROJ_NAME	CHAR	8		N. NULL		
DSF_MEASURE	D_ID	NUMBER	10, 0	PK	N. NULL		
	STATUS_CODE	CHAR	10	PK	N. NULL		
	MEASURE_CODE	CHAR	10	PK	N. NULL		
	MEASURE_VALUE	NUMBER	5, 0		N. NULL		
DSF_TARGET	D_ID	NUMBER	10, 0	PK	N. NUILL		
	STATUS_CODE	CHAR	10	PK	N. NULL		
	TARGET_CODE	CHAR	10	PK	N. NULL		
	TARGET_VALUE	NUMBER	5, 0		N. NULL		
DUMMY	HIDDEN	CHAR	1				
PERSONNEL	PROG_ID	NUMBER	5, 0	PK	N. NULL	U. INDEX	
	FORM_NAME	CHAR	15		N. NULL		
	FULL NAME	CHAR	30				
	DATE_ENTRY	DATE	9		N. NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (28 of 30)

PC Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
PROJECT	PROJ_NAME	CHAR	8	PK	N. NULL	U. INDEX	
	PROJ_NO	NUMBER	3, 0		N. NULL	U. INDEX	
	PROJ_TYPE	CHAR	10				
	ACTIVE_STATUS	CHAR	10				
PROJ_DSF	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX	
	PROG_ID	NUMBER	5, 0		N. NULL		
	FORM_NO	CHAR	6		N. NULL	U. INDEX	
	STATUS	CHAR	10		N. NULL		
	FORM_TYPE	CHAR	6		N. NULL		
	D_ID	NUMBER	10, 0		N. NULL	U. INDEX	
SEQNO	TABLE_NAME	CHAR	30	PK	N. NULL	U. INDEX	
	FIELD_NAME	CHAR	30	PK	N. NULL	U. INDEX	
	MAXSEQNO	NUMBER	10, 0		N. NULL		
TABLE_PRIVLEGE	TABLE_NAME	CHAR	40	PK	N. NULL	U. INDEX	
	USER_CLASS	CHAR	20	PK	N. NULL	U. INDEX	
	SELECT_PRIV	CHAR	1				
	INSERT_PRIV	CHAR	1				

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (29 of 30)

PC Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
TABLE_PRIVLEGE (CONT'D)	UPDATE_PRIV	CHAR	1				
	DELETE_PRIV	CHAR	1				
	ALTER_PRIV	CHAR	1				
	INDEX_PRIV	CHAR	1				
TEMP_DSF	PROJ_NO	NUMBER	3, 0				
	CODE	CHAR	10				
	PROG_ID	NUMBER	5, 0				
	SUB_DATE	DATE	9				
	D_ID	NUMBER	10, 0				
	VALUE	NUMBER	5, 0				
USER_CLASS	ORA_USER_ID	CHAR	20	PK	N. NULL	U. INDEX	
	USER_CLASS	CHAR	20		N. NULL		
USER_CLASS_ ACCESS	USER_CLASS	CHAR	20	PK	N. NULL	U. INDEX	
	ACCESS_TYPE	CHAR	10	PK	N. NULL	U. INDEX	
VALIDATION	F_NAME	CHAR	20	PK	N. NULL	U. INDEX	
	CODE	CHAR	10	PK	N. NULL	U. INDEX	
	VALUE	CHAR	75		N. NULL		

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

Table 4-2. SEL Database Tables and Views—Technical Specifications (30 of 30)

PC Tables

Table or View Name	Column Name	Type	Width	Key[1]	Nulls[2]	Indexed[3]	Underlying Table Name
VAL_DSF_MEASURE	CODE	CHAR	10		N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_DSF_STATUS	CODE	CHAR	10		N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION
VAL_DSF_TARGET	CODE	CHAR	10		N. NULL		VALIDATION
	VALUE	CHAR	75		N. NULL		VALIDATION

[1]PK = PRIMARY KEY

[2]N. NULL = NOT NULL

[3]U. INDEX = UNIQUE INDEX

4.2 RELATIONSHIPS AND CONSTRAINTS AMONG DATABASE TABLES

The SEL database is composed of two classes of information: the software engineering data itself, and the information describing those data and defining their organization within the database. The software engineering data are discussed in Sections 2 and 3. The descriptive and organizational information stored in various tables and referred to from here on as system support data are further described in this section.

4.2.1 Relationships Among Tables

In the SEL database, certain tables have relational dependencies among them. These dependencies among tables are important and need to be observed, especially when insert, update, or delete operations are performed. In a relationship, tables share common values existing in one or more columns of each table. For example, table PROJECT and table PROJ_SUB both share the same values of project number. When project data are first entered in the database, a record containing the project name, project type, and project status is created in the PROJECT table. A unique project number is also assigned and stored in the same record. As the rest of the project data are collected, they are stored in various tables. The relationship between these tables and the PROJECT table is defined through the project number column. (See Figure 1-1 for an example of this relationship between the PROJECT and PROJ_SUB tables.)

Figures 4-1 through 4-3 depict these relationships and represent them as tree structures. Figure 4-1 shows the relationships among project related data. Figure 4-2 shows the relationships among DAMSEL support tables. Figure 4-3 shows the relationships involving project-independent data.

In these figures, each tree is a logical entity of related tables. The name shown within each block is a table name. The top node in each tree is the parent node, and the others are dependent (child) nodes. Each dependent node occurrence in the tree must have a record in its parent. For example, each record existing in table SUBSYSTEM that contains detailed subsystem information must first have been created in the PROJ_SUB table, since the record in the PROJ_SUB table contains the vital information—the project number and the subsystem prefix. The name(s) shown at the upper left corner of each block corresponds to the field name that links these tables together and can be used as a joining column. For example, field COM_NO can be specified in a WHERE clause for joining tables SUB_COM and COM PURPOSE. If the common columns in both the parent and child tables have the same name, only one name is shown. Otherwise, both column names from these tables are shown and the notation "=" is used to show that they share common values. The left-hand side of the equality is the column name from the parent table; the right-hand side is the column name from the child table. For example, to join tables EFF PROJ and EFF_ACT in a SQL SELECT statement, the joining columns are P_ID from EFF_PROJ and EFF_ID from EFF_ACT.

The relationships between data elements and tables are described in detail in Reference 2. However, some of these relationships are worth mentioning here so that the reader can

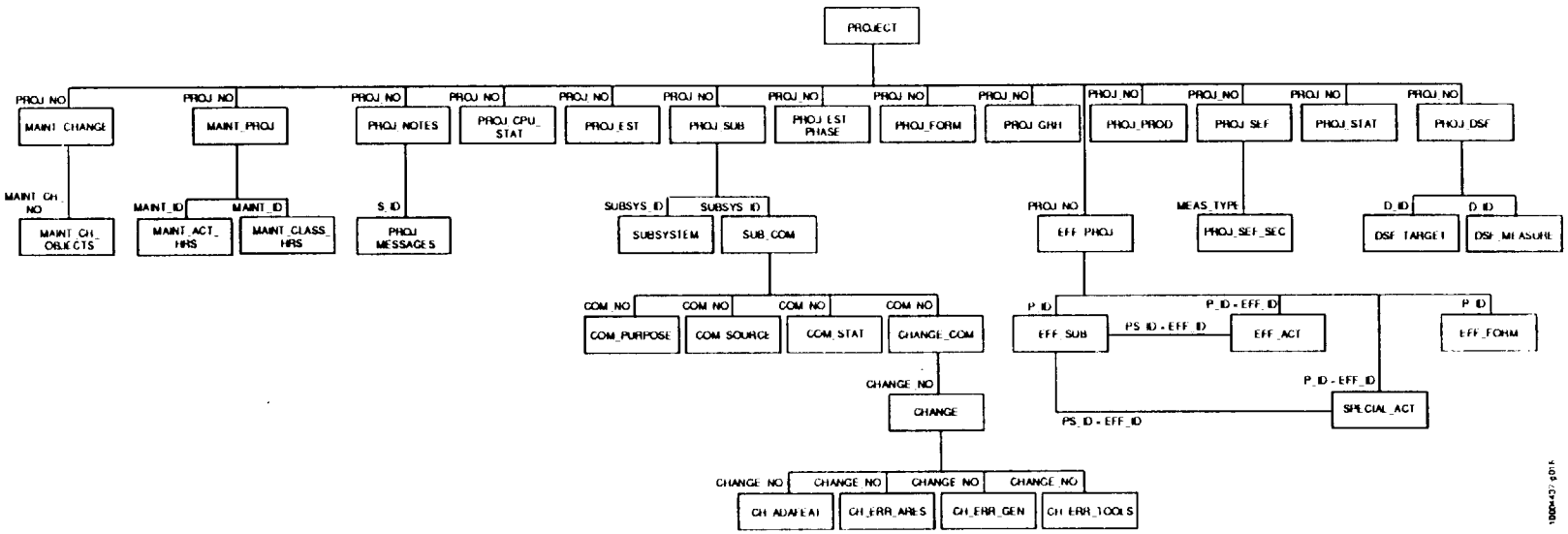


Figure 4-1. Relationships Among Project-Related Tables

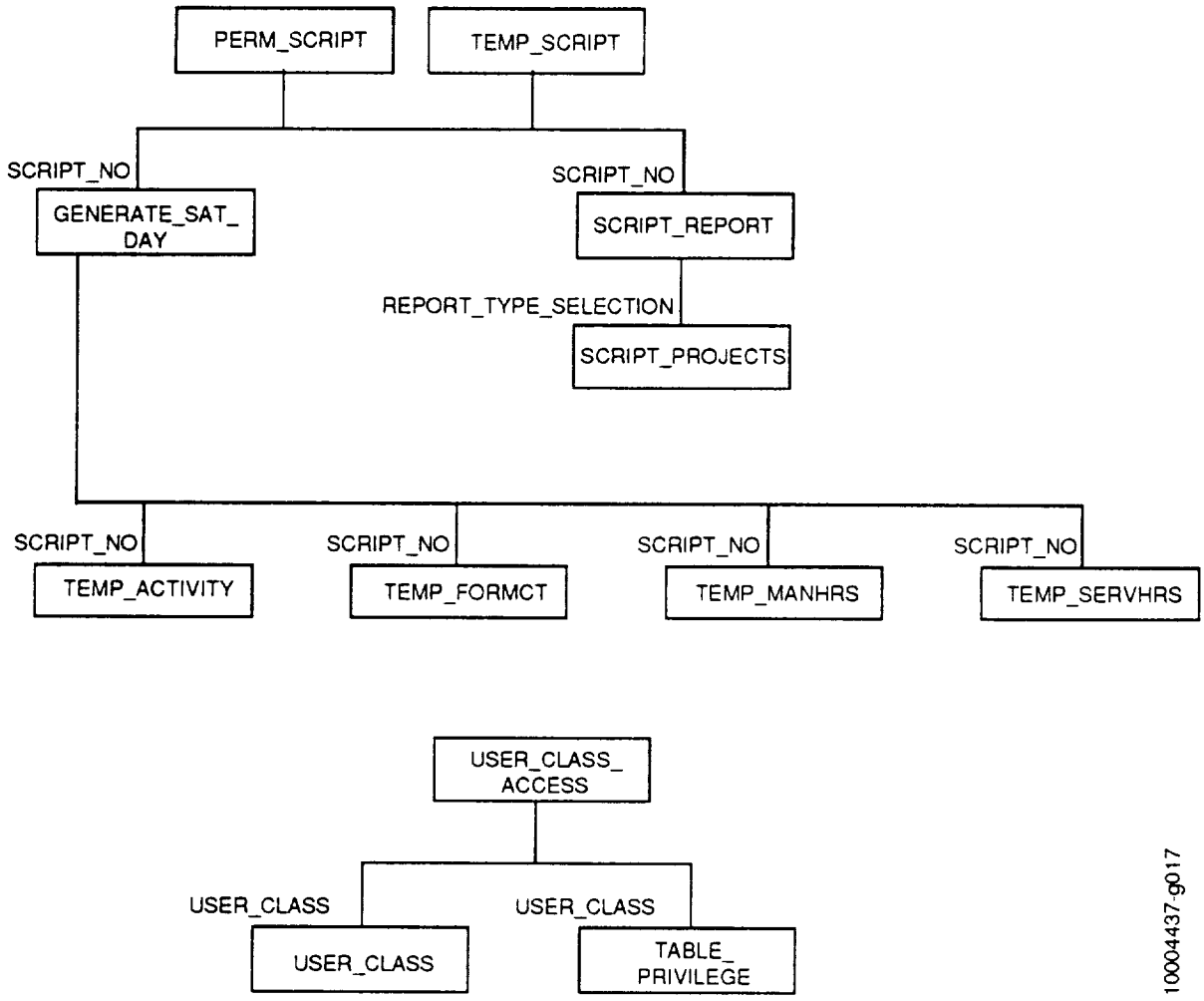


Figure 4-2. Relationships Among DAMSEL Support Tables

understand how the data are logically divided and stored in the database. Observe that the data elements that compose each of the major data groups presented in Section 2 may reside in one or more tables, depending on the number of occurrences of a particular data element. For example, consider the component information within the structure and size data group. For each component of a project, all component-related data, such as origin, creation date, type, etc., reside in the COM_SOURCE table, with the exception of the component purposes. These reside in the COM_PURPOSE table because one component can have multiple purposes. This logical partitioning of data was performed during the database design process to ensure data integrity and minimize data redundancy.

For the same reasons, staff hours information within the resource usage data group resides in different tables. Regular activity hours for all projects reside in the EFF_ACT table. The data elements required for retrieving project-related activity hours, such as project and programmer IDs, are stored in the EFF_PROJ table. Additional data elements required for retrieving

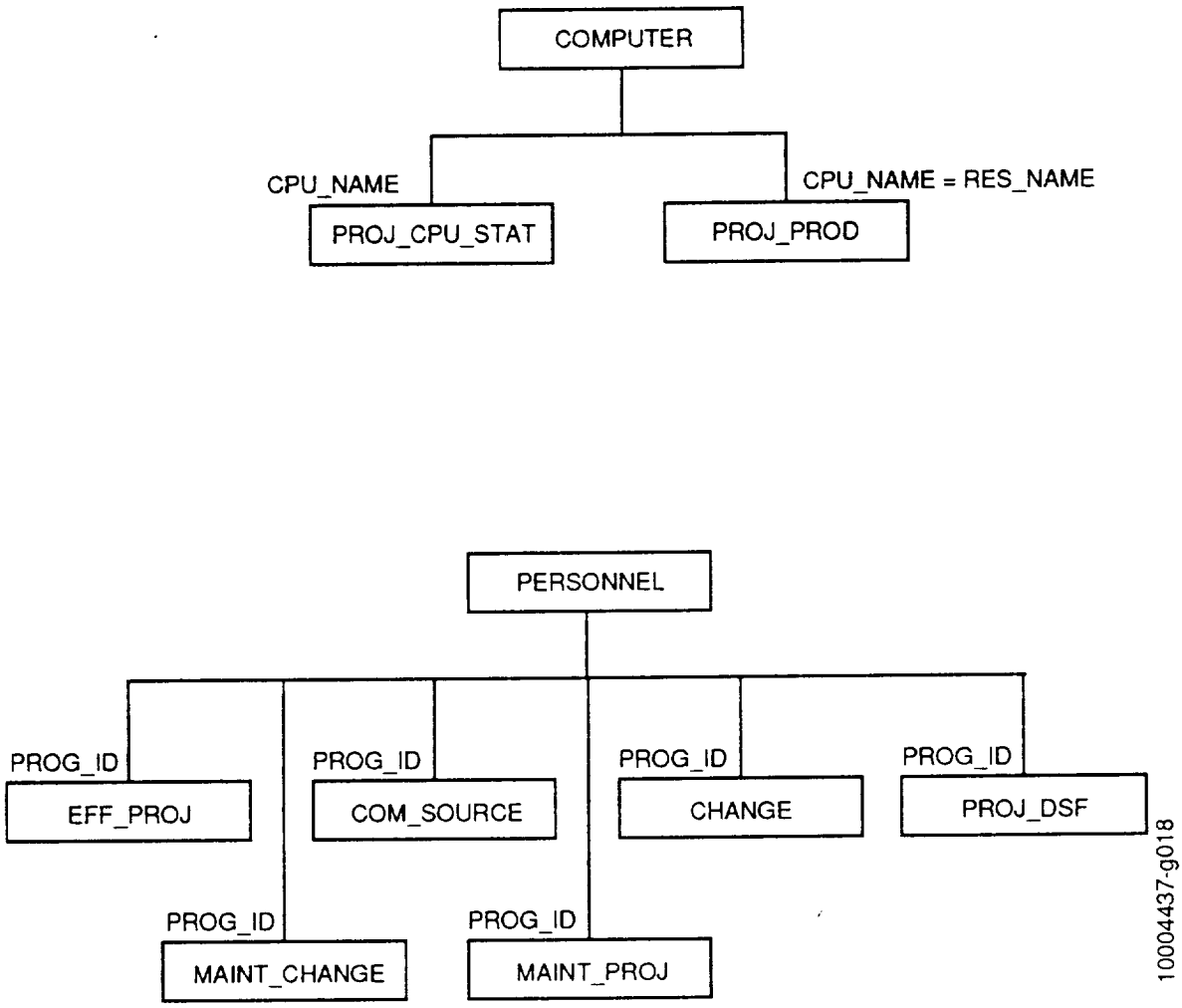


Figure 4-3. Relationships Involving Project-Independent Data

subsystem-related hours, such as subsystem prefixes, are stored in the EFF_SUB table. Using this arrangement can minimize data redundancy. As mentioned in Section 2, many projects do not have subsystem-related activity hours. Thus, depending on the project, the activity hours may be retrieved from the EFF_ACT table by directly joining it with the EFF_PROJ table, or via the EFF_SUB table. These relationships are depicted as connected lines in Figure 4-1.

As for staff hours recorded for projects using cleanroom methodology, they can be retrieved in one of two ways: as cleanroom PRF activity hours or as regular PRF activity hours. To retrieve hours under cleanroom PRF activities, join the EFF_ACT table with the EFF_PROJ table and specify the cleanroom activities. The cleanroom PRF activities are provided in Appendix A of this document or can be viewed in the database by selecting codes and values from the view VAL_CL_ACTIVITY. To retrieve hours under the regular PRF activities, join

. the view V_CLEANROOM_ACT with table EFF_PROJ. The mapping between the cleanroom PRF activities and the regular PRF activities is as follows:

Cleanroom PRF Activity/Code	Regular PRF Activity/Code
Predesign (CLPREDES)	Predesign (PREDES)
Create design (CLCREDES)	Create design (CREDES)
Verify/Review design (CLVEREVDES)	Read/Review design (RDREVDES)
Write code (CLWRCODE)	Write code (WRCODE)
Read/Review code (CLRDREVCOD)	Read/Review code (RDREVCOD)
Pretest + Independent test (CLPRETEST + CLINDTEST)	Integration test (INTTEST)
Response to SFR (CLRESPSFR)	Debugging (DEBUG)
Acceptance test (CLACCTEST)	Acceptance test (ACCTEST)
Other (CLOTHER)	Other (OTHER)

In addition, some of the tables are used as connectors to relate data items that reside in different tables. For example, consider the CHANGE_COM table within the change data group. It does not contain any SEL forms data. It only contains two surrogate key fields, change number and component number. The fields in this table can be used to connect the change data with the size and structure data (i.e., project and subsystem data items stored in various tables). Other tables, such as PROJ_SUB and SUB_COM, have a functionality similar to the CHANGE_COM table.

4.2.2 Descriptions of Support Data Tables

The tables described in this section do not contain software engineering data. Rather, they are used to store data that are internal to the database structure and to store data that are used by the database operational software.

CLOSE_COF

This table is used during project closeout for verifying the accuracy and completeness of a project's COFs. This temporary table is cleared, populated with all the component information for the specified project, queried, and cleared again.

CLOSE_COM_NO_ORIGIN

This table is used during project closeout for assigning a final "origin" category to each component. For most components the final "origin" is the same as the COF origin. However, any component with a COF origin of "Old and Unchanged" will be assigned a final "origin" of slightly modified if any CRFs were submitted for that component.

CLOSE_CRF

This table is used during project closeout for verifying the accuracy and completeness of a project's CRFs. This temporary table is cleared, populated with all the change information for the specified project, queried, and cleared again.

CLOSE_CRF_ERR

This table is used during project closeout for verifying the accuracy and completeness of a project's CRFs with a change type of error correction (ERRCO). This temporary table is cleared, populated with all the information about changes due to errors for the specific project, queried, and cleared again.

CRF_TEMP_CHANGE_COM

This table is used by the DAMSEL CRF data entry programs CRF_INSERT, CRF_UPDATE, and CRF_QA. It contains the component information associated with the current CRF form. The information is uniquely identified with a USER_ID, which is actually the SESSIONID of the current user.

DUMMY

This table is used by DAMSEL data entry programs. It is updated with null values during data entry to invoke, or trigger, certain sequences of operations to be performed.

GENERATE_SAT_DAY

This table is used in generating DAMSEL reports. It stores all the Saturday dates for reports that display weekly information. Once the dates are used by a report, the corresponding entries in this table are then deleted.

PC_SEQNO

This table is used by the DAMSEL DSF data entry software. The PROJ_DSF table contains two columns that are system-generated numeric IDs: D_ID and FORM_NO. The PC_SEQNO table stores the maximum value that already exists in PROJ_DSF for each of these fields.

PERM_SCRIPT

This table is used in generating DAMSEL reports. It contains header information about the permanent report scripts. A report script is built during interactive report selection via DAMSEL. A script is identified by a script number and its owner's ORACLE USER_ID.

REP_CODES

This table is used as a look-up table by the DAMSEL menus and screens. It contains all the possible report types, report titles, report codes, and project selection criteria. Each entry in the table contains a unique code and a descriptive value. The codes are stored, but the values are displayed on the screens so that users will understand the contents of a report script.

SCRIPT_PROJECTS

This table is used in generating DAMSEL reports. It stores the names of the projects that are entered by a user for multiple-project reports with a REPORT_TYPE_SELECTION (in table SCRIPT_REPORT) of "LIST." The entries that are created for temporary scripts are deleted once the report has been generated; the entries for permanent scripts are stored until the script owner deletes the script.

SCRIPT_REPORT

This table is used in generating DAMSEL reports. It contains the definitions of both temporary and permanent scripts. The following information is stored for each report in a script: the report type (e.g., single-project or multiple-project); the report code, which identifies the report; the project(s) to be included in the report; and the report sequence number, which identifies the location of the report within the script.

SEQNO

This table is used by DAMSEL data entry programs. It stores the maximum values already used of all the system-generated IDs in the database. The following columns are system-generated IDs :

Table Name	Column Name
EFF_PROJ	P_ID
EFF_SUB	PS_ID
MAINT_PROJ	MAINT_ID
PERM_SCRIPT	SCRIPT_NO
PERSONNEL	PROG_ID
PROJECT	PROJ_NO
PROJ_NOTES	S_ID
PROJ_SUB	SUBSY_ID
SUB_COM	COM_NO
TEMP_SCRIPT	SCRIPT_NO

TABLE_PRIVILEGE

This table is used in enrolling DAMSEL users. It defines the access privileges that each user class may be granted for each table in the database. The valid privileges are select, insert, update, delete, alter table structure, and create indices.

TEMP_ACTIVITY

This table is used for producing the DAMSEL Programmer Activity Hours reports. It contains all of the possible activities for each week the project has been in a development

phase. For each activity and week, the total number of hours worked on the project is stored. To populate this table, the GENERATE_SAT_DAY table must first be populated with the correct Saturday dates.

TEMP_FORMCT

This table is used for producing the DAMSEL Project Form Counts reports. It contains the total number of CRFs, COFs, and SPFs that have been entered since the project has been in a development phase. For each form type and week, the total number of forms entered is stored.

TEMP_MANHRS

This table is used for producing the DAMSEL Manpower Hours reports. It contains all of the programmer names for each week the project has been in a development phase. For each programmer and week, the total number of hours worked is stored. To populate this table, the GENERATE SAT DAY table must first be populated with the correct Saturday dates.

TEMP_SCRIPT

This table is used in generating DAMSEL reports. It contains header information about the temporary report scripts that are created by each user during an interactive session. The script owner, his/her process ID, the script status, and other script-related information are stored in this table. The scripts are identified by script numbers.

TEMP SERVHRS

This table is used for producing the DAMSEL Services Hours reports. It contains all of the support names for each week the project has been in a development phase. For each support name and week, the total number of hours worked is stored. To populate this table, the GENERATE_SAT_DAY table must first be populated with the correct Saturday dates.

T_COM_STAT

This table is used during project closeout to load the COM_STAT table. Records are loaded from a flat file into T_COM_STAT via SQL*Loader. The T_COM_STAT rows and SQL*Loader output are then verified by SEL personnel before the rows are inserted into COM_STAT.

USER_CLASS

This table is used in enrolling DAMSEL users. It contains all users' ORACLE user IDs and their user class specifications. Currently, there are five types of user classes: general user, librarian, quality assurance (QA), SEL database administrator (DBA), and system maintenance user.

USER_CLASS_ACCESS

This table is used in enrolling DAMSEL users. For each user class specification, the types of functional access permitted are stored in this table. The current valid types of access are BACKUP, DBA, DELETE, DISTAPE, FORM, GENERAL, IMPORT, INSERT, QA, QUERY, REPORT, RESTORE, UPDATE, UPDOWN, AND VIEW.

VALIDATION

This table stores all the codes and their corresponding detailed descriptions used by various tables throughout the database. (Appendix A provides a complete list of all the codes and their descriptions.) Fields that use coded values are listed below.

Table or View Name	Field Name
CHANGE	CH_TYPE
CHANGE	EFF_COM_CH
CHANGE	EFF_ISO_CH
CHANGE	STATUS
CH_ADAFEAT	ADA_FEATURE
CH_ERR_ARES	ERR_ARES
CH_ERR_GEN	ERR_ACAUSE
CH_ERR_GEN	ERR_CLASS
CH_ERR_GEN	ERR_SOURCE
CH_ERR_TOOLS	ERR_TOOLS
COM_PURPOSE	PURPOSE
COM_SOURCE	COM_TYPE
COM_SOURCE	ORI_TYPE
COM_SOURCE	STATUS
COM_STAT	FINAL_ORIGIN_CAT
DSF_MEASURE	MEASURE CODE
DSF_MEASURE	STATUS_CODE
DSF_TARGET	STATUS_CODE
DSF_TARGET	TARGET_CODE
EFF_ACT	ACTIVITY
EFF_FORM	STATUS
MAINT_ACT_HRS	MAINT_ACT
MAINT_CHANGE	CH_CAUSE
MAINT_CHANGE	CH_CLASS
MAINT_CHANGE	MAINT_CH_TYPE
MAINT_CHANGE	MAINT_COM_CH
MAINT_CHANGE	MAINT_ISO_CH
MAINT_CH_OBJECTS	CH_OBJECT

Table or View Name	Field Name
MAINT_CLASS_HRS	MAINT_CLASS
PROJECT	ACTIVE_STATUS
PROJECT	PROJ_TYPE
PROJ_EST_PHASE	PHASE_CO
PROJ_FORM	STATUS
PROJ_NOTES	NOTE_TYPE
PROJ_SEF	MEAS_TYPE
PROJ_SEF_SEC	SECOND_L
SPECIAL_ACT	SP_ACTIVITY
SUBSYSTEM	FUNCTION
VAL_CL_ACTIVITY	CL_ACTIVITY
VAL_DATA_AVAIL	DATA_AVAIL
VAL_QA_STATUS	QA_STATUS

4.2.3 Database Constraints

Various constraints are associated with the database. Constraints are defined to ensure that the database contains only accurate and consistent data and to protect the data against unauthorized or accidental alterations. In the SEL database environment, constraints are identified as access constraints or data integrity constraints. Access constraints are associated with each user class and are defined as follows:

- General user—Has read access to all data
- Data librarian—Has read, write, and update access to the form-related data
- QA—Has read and update access to certain form related data
- DBA—Has read, write, and update access to all data
- System maintenance—Has read access to all data, and read, write, and update access to system support data

Data integrity constraints are applied to all insertions to, deletions from, and updates of the database. Table 4-3 describes these constraints. They are used not only in SQL queries, but also in the DAMSEL data entry software. Table 4-3 lists only the database tables that have constraints. In addition to these constraints, field EFF_ID in table EFF_ACT and table SPECIAL_ACT contains values from both the P_ID field (in table EFF_PROJ) and the PS_ID field (in table EFF_SUB). This constraint is accommodated by assigning mutually exclusive values for P_ID and PS_ID.

4.3 MAPPING THE CONCEPTUAL VIEW TO THE LOGICAL VIEW

This section presents a schema, shown in Table 4-4 (at the end of the section), that maps both the conceptual and the data collection views of the SEL data described in Sections 2 and 3 to a unified logical view. The schema is intended to provide general users who would like to retrieve data using SQL queries with more detailed information on how to get to the desired data. By using this schema, along with the specific instructions on how to access SQL*Plus in the SEL database environment (provided in Section 5.3), general users can set up their own queries to look at the data in their own specific ways.

Table 4-4 lists all the reference IDs used in Sections 2 and 3 that identify the data items in the database and presents the name of the table and the column where that data item is stored. This table is ordered by target table and target column.

Required access information, needed to obtain a particular piece of data, is also provided for each reference ID. Under the columns "TARGET TABLE" and "TARGET COLUMN" is the table/field from which data are being retrieved. For example, to retrieve the activity hours for a particular programmer (see Table 4-4, under TARGET TABLE EFF_ACT and TARGET COLUMN ACT_HR), the project name, the programmer name, and the week ending date on the PRF must be provided before the appropriate activity hours can be retrieved.

Under the heading "Access Path," there is a graph-like diagram showing the access path that a SQL query may traverse to retrieve the desired data. The path shown is just one of the many possible ways to get to the data; other paths can be used to achieve the same result. In each access path, the names within square brackets [ ] represent column names. The names with no brackets around them represent table names. The arrows point to either an intermediate table or the final target column. The name of each target field that stores coded values is followed by the keywords "*CODED FIELD." The codes and their descriptions are explained in Appendix A. In addition, symbol "!=" means not equal to and MAX means the maximum value of the column that follows.

Using the access paths in Table 4-4, the corresponding SQL queries can be formulated easily. The following three examples demonstrate how to interpret the access path diagrams. They also show that some of the access paths may retrieve a single record from a target table and others may retrieve multiple records. In the first example, the access path will return one record if one subsystem exists for the specified project; multiple records if more than one subsystem exists; or null if no subsystems exist. In the second example, the access path will return a single record that contains the creation date for the component specified by the user. However, this access path can be modified to retrieve all the creation dates for all components in a particular subsystem within a particular project. This can be accomplished by not specifying the component name in the SQL query. The third example retrieves the same information as example 2. The difference is that a view is joined to one table to simplify the query and eliminate the need to join four tables.

Table 4-3. Constraints on Database Tables (1 of 6)

Table	Constraint
CHANGE	THE CRF FORM NUMBER (CHANGE_NO) MUST BE UNIQUE WITHIN THIS TABLE.
	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE EFFORT TO ISOLATE CHANGES CODE (EFF_ISO_CH) MUST EXIST IN THE VAL_ISO_CH VIEW.
	THE EFFORT TO IMPLEMENT CHANGES CODE (EFF_COM_CH) MUST EXIST IN THE VAL_COM_CH VIEW.
	THE TYPE OF CHANGE (CH_TYPE) MUST EXIST IN THE VAL_CH_TYPE VIEW.
	THE FORM TYPE (FORM_TYPE) MUST EQUAL 'CRF'.
	THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW.
CHANGE_COM	THE CRF FORM NUMBER (CHANGE_NO) MUST EXIST IN THE CHANGE TABLE.
	THE COMPONENT NUMBER (COM_NO) MUST EXIST IN THE SUB_COM TABLE.
CH_ADAFEAT	THE CRF FORM NUMBER (CHANGE_NO) MUST EXIST IN THE CHANGE TABLE, AND THE FLAG INDICATING WHETHER THE USE OF ADA CONTRIBUTED TO THE CHANGE (EFF_ADA) IN THE CHANGE TABLE MUST EQUAL 'Y' FOR THAT CHANGE.
	THE ADA FEATURE CODE (ADA_FEATURE) MUST EXIST IN THE VAL_ADA_FEATURE VIEW.
CH_ERR_ARES	THE CRF FORM NUMBER (CHANGE_NO) MUST EXIST IN THE CHANGE TABLE, THE TYPE OF CHANGE (CH_TYPE) IN THE CHANGE TABLE MUST EQUAL 'ERRCO' FOR THAT CHANGE, AND EFF_ADA MUST EQUAL 'Y'
	THE CODE REPRESENTING THE RESOURCE NEEDED TO CORRECT AN ADA ERROR (ERR_ARES) MUST EXIST IN THE VAL_ERR_ARES VIEW.
CH_ERR_GEN	THE CRF FORM NUMBER (CHANGE_NO) MUST EXIST IN THE CHANGE TABLE, AND THE TYPE OF CHANGE (CH_TYPE) IN THE CHANGE TABLE MUST EQUAL 'ERRCO' FOR THAT CHANGE.
	THE SOURCE OF ERROR CODE (ERR_SOURCE) MUST EXIST IN THE VAL_ERR_SOURCE VIEW.
	THE CLASS OF ERROR CODE (ERR_CLASS) MUST EXIST IN THE VAL_ERR_CLASS VIEW.
	THE CODE FOR THE CAUSE OF AN ERROR INVOLVING ADA (ERR_ACAUSE) MUST EXIST IN THE VAL_ERR_ACAUSE VIEW.
CH_ERR_TOOLS	THE CRF FORM NUMBER (CHANGE_NO) MUST EXIST IN THE CHANGE TABLE, THE TYPE OF CHANGE (CH_TYPE) IN THE CHANGE TABLE MUST EQUAL 'ERRCO' FOR THAT CHANGE, AND EFF_ADA MUST EQUAL 'Y'.
	THE CODE FOR ADA TOOLS AIDING IN THE DETECTION OR CORRECTION OF AN ERROR (ERR_TOOLS) MUST EXIST IN THE VAL_ERR_TOOLS VIEW.
COMPUTER	THE COMPUTER NAME (CPU_NAME) MUST BE UNIQUE WITHIN THIS TABLE.
COM_PURPOSE	THE COMPONENT NUMBER (COM_NO) MUST EXIST IN THE SUB_COM TABLE.
	THE COMPONENT PURPOSE (PURPOSE) MUST EXIST IN VAL_COM_PURPOSE VIEW.
COM_SOURCE	THE COMPONENT NUMBER (COM_NO) MUST EXIST IN THE SUB_COM TABLE.
	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE COF NUMBER (FORM_NO) MUST BE UNIQUE WITHIN THIS TABLE.
	THE FORM TYPE (FORM_TYPE) MUST EQUAL 'COF'.

Table 4-3. Constraints on Database Tables (2 of 6)

Table	Constraint
COM_SOURCE (CONT'D)	THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW.
	THE ORIGIN OF A COMPONENT CODE (ORI_TYPE) MUST EXIST IN THE VAL_ORI_TYPE VIEW.
	THE COMPONENT TYPE CODE (COM_TYPE) MUST EXIST IN THE VAL_COM_TYPE VIEW.
COM_STAT	THE COMPONENT NUMBER (COM_NO) MUST EXIST IN THE SUB_COM TABLE.
CRF_TEMP_ CHANGE_COM	THE SUBSYSTEM PREFIX (SUB_PRE) MUST EXIST IN THE PROJ_SUB TABLE.
	THE COMPONENT NAME (COM_NAME) MUST EXIST IN THE V_PROJ_COM VIEW.
	THE COMPONENT NUMBER (COM_NO) MUST EXIST IN THE V_PROJ_COM VIEW.
DSF_MEASURE	THE D_ID MUST EXIST IN THE PROJ_DSF TABLE.
	THE DSF STATUS CODE (STATUS_CODE) MUST EXIST IN THE VAL_DSF_STATUS VIEW.
	THE DSF MEASURE CODE (MEASURE_CODE) MUST EXIST IN THE VAL_DSF_MEASURE VIEW.
DSF_TARGET	THE D_ID MUST EXIST IN THE PROJ_DSF TABLE.
	THE DSF STATUS CODE (STATUS_CODE) MUST EXIST IN THE VAL_DSF_STATUS VIEW.
	THE DSF TARGET CODE (TARGET_CODE) MUST EXIST IN THE VAL_DSF TARGET VIEW.
EFF_ACT	THE EFF_ID MUST EXIST IN THE EFF_SUB (AS PS_ID) OR IN THE EFF_PROJ (AS P_ID) TABLE.
	THE ACTIVITY CODE (ACTIVITY) MUST EXIST IN THE VAL_ACTIVITY VIEW.
EFF_FORM	THE P_ID MUST EXIST IN THE EFF_PROJ TABLE.
	THE FORM TYPE (FORM_TYPE) MUST BE 'CLPRF', 'PRF', OR 'SPF'.
	THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW.
EFF_PROJ	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE SUBMISSION DATE (SUB_DATE) MUST BE A VALID FRIDAY DATE.
	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE P_ID MUST BE UNIQUE WITHIN THIS TABLE.
EFF_SUB	THE P_ID MUST EXIST IN THE EFF_PROJ TABLE.
	THE SUBSYSTEM PREFIX (SUB_PRE) MUST EXIST IN THE PROJ_SUB TABLE.
	THE PS_ID MUST BE UNIQUE WITHIN THIS TABLE.
GENERATE_ SAT_DAY	THE REPORT SCRIPT NUMBER (SCRIPT_NO) MUST EXIST IN THE TEMP_SCRIPT TABLE.
	THE DATE (SAT_DAY) MUST BE A VALID SATURDAY DATE.
MAINT_ACT_ HRS	THE MAINT_ID MUST BE IN THE MAINT_PROJ TABLE.
	THE MAINTENANCE ACTIVITY CODE (MAINT_ACT) MUST EXIST IN THE VAL_MAINT_ACT VIEW.
	THE COMBINATION OF THE MAINT_ID AND MAINT_ACT MUST BE UNIQUE.
MAINT_CHANGE	THE MAINTENANCE CHANGE NUMBER (MAINT_CH_NO) MUST BE UNIQUE WITHIN THIS TABLE.

Table 4-3. Constraints on Database Tables (3 of 6)

Table	Constraint
MAINT_CHANGE (CONT'D)	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW.
	THE FORM TYPE (FORM_TYPE) MUST BE 'MCRF'.
	THE TYPE OF CHANGE (MAINT_CH_TYPE) MUST EXIST IN THE VAL_MAINT_CH_TYPE VIEW.
	THE CAUSE OF CHANGE (CH_CAUSE) MUST EXIST IN THE VAL_CH_TYPE VIEW.
	THE EFFORT TO ISOLATE CHANGES CODE (MAINT_ISO_CH) MUST EXIST IN THE VAL_MAINT_ISO_CH.
	THE EFFORT TO IMPLEMENT CHANGES CODE (MAINT_COM_CH) MUST EXIST IN THE VAL_MAINT_COM_CH VIEW.
	THE CHARACTERISTIC OF CHANGE (CH_CLASS) MUST EXIST IN THE VAL_CH_CLASS VIEW.
MAINT_CH_ OBJECTS	THE MAINTENANCE CHANGE NUMBER (MAINT_CH_NO) MUST EXIST IN THE MAINT_CHANGE TABLE.
	THE CHANGE OBJECTS (CH_OBJECT) MUST EXIST IN THE VAL_CH_OBJECT VIEW.
MAINT_CLASS_HRS	THE MAINT_ID MUST BE IN THE MAINT_PROJ TABLE.
	THE CLASS OF MAINTENANCE (MAINT_CLASS) MUST EXIST IN THE VAL_MAINT_CLASS VIEW.
	THE COMBINATION OF THE MAINT_ID AND MAINT_CLASS MUST BE UNIQUE.
MAINT_PROJ	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE SUBMISSION DATE (SUB_DATE) MUST BE A VALID FRIDAY DATE.
	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE MAINT_ID MUST BE UNIQUE WITHIN THIS TABLE.
	THE FORM TYPE (FORM_TYPE) MUST BE 'WMEF'.
	THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW.
PC_SEQNO	THE TABLE NAME (TABLE_NAME) MUST EXIST IN THE DATABASE.
	THE FIELD NAME (FIELD_NAME) MUST EXIST IN THAT PARTICULAR TABLE.
PERM_SCRIPT	THE ORACLE USER ID (ORA_USER) MUST EXIST IN THE USER_CLASS TABLE.
	THE SCRIPT NUMBER (SCRIPT_NO) MUST BE UNIQUE WITHIN THIS TABLE.
	THE OUTPUT DESTINATION (OUT_ROUTING) MUST BE 'P' FOR PRINTER OR 'F' FOR FILE.
	THE OUTPUT FILE NAME (OUT_FILE) MUST BE ENTERED IF OUT_ROUTING EQUALS 'F'.
PERSONNEL	THE ABBREVIATED NAME USED ON FORMS (FORM_NAME) MUST BE UNIQUE WITHIN THIS TABLE.
	THE PROG_ID MUST BE UNIQUE WITHIN THIS TABLE.
PROJECT	THE PROJECT NAME (PROJ_NAME) MUST BE UNIQUE WITHIN THIS TABLE.
	THE PROJECT NUMBER (PROJ_NO) MUST BE UNIQUE WITHIN THIS TABLE.

Table 4-3. Constraints on Database Tables (4 of 6)

Table	Constraint
PROJ_CPU_ STAT	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE COMPUTER NAME (CPU_NAME) MUST EXIST IN THE COMPUTER TABLE.
PROJ_DSF	PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE SUBMISSION DATE (SUB_DATE) MUST BE A VALID FRIDAY DATE.
	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW.
	THE FORM TYPE (FORM_TYPE) MUST BE 'DSF'.
	THE D_ID MUST BE UNIQUE WITHIN THIS TABLE.
PROJ_EST	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
PROJ_EST_ PHASE	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE PHASE CODE (PHASE_CO) MUST EXIST IN THE VAL_PHASE VIEW.
	THE PHASE START DATE (START_DATE) AND END DATE (END_DATE) MUST BE VALID SATURDAY DATES.
PROJ_FORM	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE FORM NUMBER (FORM_NO) MUST BE UNIQUE WITHIN THIS TABLE FOR A PARTICULAR FORM TYPE.
	THE FORM TYPE (FORM_TYPE) MUST BE 'PCSF', 'PEF', 'SEF', 'SPF'.
	THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW.
PROJ_GRH	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE SUBMISSION DATE (SUB_DATE) MUST BE A VALID FRIDAY DATE.
PROJ_ MESSAGES	THE S_ID MUST EXIST IN THE PROJ_NOTES TABLE.
PROJ_NOTES	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE MESSAGE TYPE (NOTE_TYPE) MUST EXIST IN THE VAL_NOTE_TYPE VIEW.
	THE S_ID MUST BE UNIQUE WITHIN THIS TABLE.
PROJ_PROD	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE SUBMISSION DATE (SUB_DATE) MUST BE A VALID FRIDAY DATE.
	THE COMPUTER NAME (RES_NAME) MUST EXIST IN THE COMPUTER TABLE.
PROJ_SEF	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE SUBJECTIVE EVALUATION MEASUREMENT (MEAS_TYPE) MUST EXIST IN THE VAL_MEAS_TYPE VIEW.
PROJ_SEF_SEC	THE SUBJECTIVE EVALUATION MEASUREMENT (MEAS_TYPE) AND THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJ_SEF TABLE.
	THE SECONDARY-LEVEL INFORMATION MEASUREMENT CODES (SECOND_L) MUST EXIST IN THE VAL_SECOND_L VIEW.
PROJ_STAT	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
PROJ_SUB	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE SUBSYSTEM PREFIX (SUB_PRE) MUST BE UNIQUE WITHIN THIS TABLE FOR A PARTICULAR PROJ_NO.

Table 4-3. Constraints on Database Tables (5 of 6)

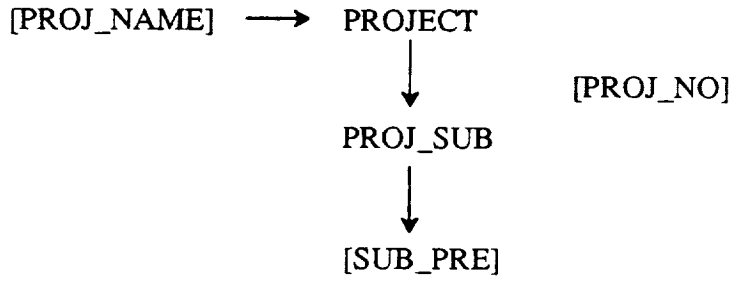
Table	Constraint
PROJ_SUB (CONT'D)	THE SUBSYSTEM ID (SUBSY_ID) MUST BE UNIQUE WITHIN THIS TABLE.
SCRIPT_ PROJECTS	THE SCRIPT NUMBER (SCRIPT_NO) AND THE REPORT SEQUENCE NUMBER (REPORT_SEQ) MUST EXIST IN THE SCRIPT_REPORT TABLE.
	THE PROJECT NAME (PROJ_NAME) MUST EXIST IN THE PROJECT TABLE.
SCRIPT_ REPORT	THE SCRIPT NUMBER (SCRIPT_NO) MUST EXIST IN EITHER THE PERM_SCRIPT OR THE TEMP_SCRIPT TABLE.
	THE REPORT CODE (REPORT_CODE) MUST EXIST IN THE VAL_REPORT_CODE TABLE.
	THE REPORT TYPE CODE (REPORT_TYPE) MUST BE 'S' FOR SINGLE PROJECT REPORT, 'M' FOR MULTIPLE-PROJECT REPORT, OR 'O' FOR MISCELLANEOUS REPORT.
	IF REPORT_TYPE EQUALS 'S', THE VALID VALUES FOR REPORT_TYPE_SELECTION ARE VALID PROJECT NAMES (PROJ_NAME) IN THE PROJECT TABLE. IF REPORT_TYPE EQUALS 'M', THE VALID VALUES FOR REPORT_TYPE_SELECTION ARE 'ALL', 'ACT_DEV', 'ACT_MAINT', 'INACTIVE', AND 'LIST'. IF REPORT_TYPE EQUALS 'O', THE REPORT TYPE SELECTION IS NULL.
SEQNO	THE TABLE NAME (TABLE_NAME) MUST EXIST IN THE DATABASE.
	THE FIELD NAME (FIELD_NAME) MUST EXIST IN THAT PRTICULAR TABLE.
SPECIAL_ACT	THE EFF_ID MUST EXIST IN EITHER THE EFF_PROJ (AS P_ID) OR THE EFF_SUB (AS PS_ID) TABLE.
	THE SPECIAL ACTIVITY CODE (SP_ACTIVITY) MUST EXIST IN THE VAL_SP_ACTIVITY VIEW.
SUBSYSTEM	THE SUBSYSTEM ID (SUBSY_ID) MUST EXIST IN THE PROJ_SUB TABLE.
	THE SUBSYSTEM FUNCTION (FUNCTION) MUST EXIST IN THE VAL_S_FUNCTION VIEW.
SUB_COM	THE SUBSYSTEM ID (SUBSY_ID) MUST EXIST IN THE PROJ_SUB TABLE.
	THE COMPONENT NAME (COM_NAME) MUST BE UNIQUE WITHIN THIS TABLE FOR A PARTICULAR SUBSYSTEM.
	THE COMPONENT NUMBER (COM_NO) MUST BE UNIQUE WITHIN THIS TABLE.
TABLE_ PRIVILEGE	THE TABLE NAME (TABLE_NAME) MUST EXIST IN THE DATABASE.
	THE USER CLASS (USER_CLASS) MUST EXIST IN THE USER_CLASS TABLE.
TEMP_ACTIVITY	THE SCRIPT NUMBER (SCRIPT_NO) AND SATURDAY DATE (SAT_DAY) MUST EXIST IN THE GENERATE_SAT_DAY TABLE.
	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
TEMP_FORMCT	THE SCRIPT NUMBER (SCRIPT_NO) MUST EXIST IN THE TEMP_SCRIPT TABLE.
	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
TEMP_MANHRS	THE SCRIPT NUMBER (SCRIPT_NO) AND SATURDAY DATE (SAT_DAY) MUST EXIST IN THE GENERATE_SAT_DAY TABLE.
	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE P_ID MUST EXIST IN THE EFF_PROJ TABLE.

Table 4-3. Constraints on Database Tables (6 of 6)

Table	Constraint
TEMP SCRIPT	THE SCRIPT NUMBER (SCRIPT_NO) MUST BE UNIQUE WITHIN THIS TABLE.
	THE ORACLE USER ID (ORA_USER) MUST EXIST IN THE USER_CLASS TABLE.
	THE OUTPUT DESTINATION (OUT_ROUTING) MUST BE 'P' FOR PRINTER OR 'F' FOR FILE.
	THE OUTPUT FILE NAME (OUT_FILE) MUST BE ENTERED IF OUT_ROUTING EQUALS 'F'.
TEMP_SERVHRS	THE SCRIPT_NO AND SAT_DAY MUST EXIST IN THE GENERATE_SAT_DAY TABLE.
	THE PROGRAMMER ID (PROG ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE
	THE P_ID MUST EXIST IN THE EFF_PROJ TABLE.
USER_CLASS	THE ORACLE USER ID (ORA_USER_ID) MUST BE A VALID ORACLE USER ACCOUNT NAME.
	THE CLASS OF USER (USER_CLASS) MUST EXIST IN THE USER_CLASS_ACCESS TABLE.

Example 1

This example retrieves all the subsystem prefixes of a particular project. This access path is shown in Table 4-4 under target table PROJ_SUB and target column SUB_PRE and is as follows:



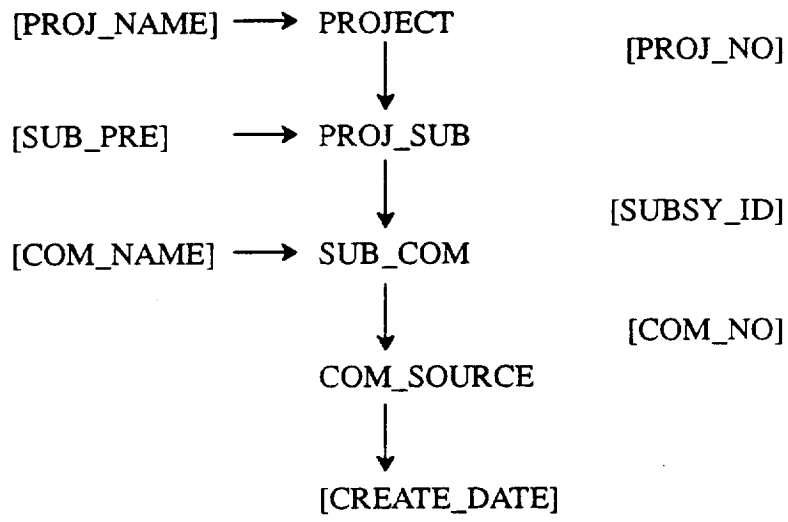
The first line in the access path shows that PROJ_NAME is the field whose value must be specified by the user to identify which project's data are to be retrieved. The down arrow between PROJECT and PROJ SUB means that the two tables are joined together by a common field, which is listed next to the arrow (PROJ_NO, in this case). The down arrow under PROJ SUB points to the target column SUB_PRE of table PROJ_SUB, where all the subsystem prefixes are stored.

SQL statement

```
SQL>    SELECT SUB_PRE FROM PROJ_SUB, PROJECT
  2     WHERE PROJ_SUB. PROJ_NO = PROJECT. PROJ_NO
  3     AND     PROJ_NAME = <user-supplied project name>;
```

Example 2

This example retrieves the date on which a component was entered into the project's controlled library. The access path for this example is shown in Table 4-4 under target table COM_SOURCE and target column CREATE_DATE and is as follows:



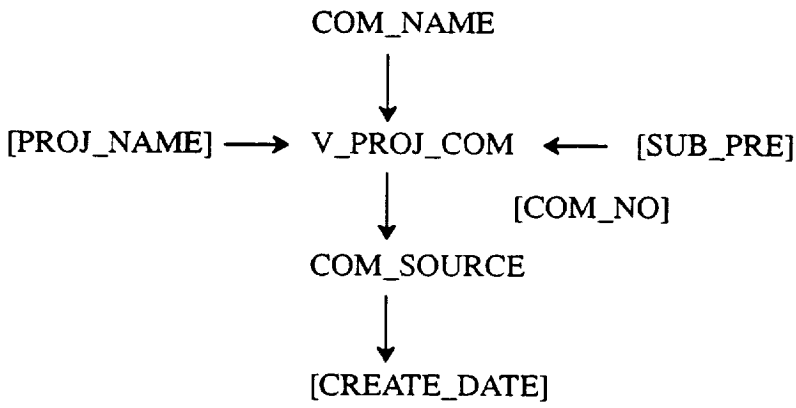
PROJ_NAME, SUB_PRE, and COM_NAME are the fields whose values must be provided by the user. Tables PROJECT and PROJ_SUB are joined on PROJ_NO; PROJ_SUB and SUB_COM are joined on SUBSY_ID; and SUB_COM and COM_SOURCE are joined on COM_NO. The result is field CREATE_DATE of the COM_SOURCE table.

SQL statement

```
SQL>   SELECT  CREATE_DATE
  2    FROM    COM_SOURCE, SUB_COM, PROJ_SUB, PROJECT
  3    WHERE  COM_SOURCE. COM_NO = SUB_COM. COM_NO
  4    AND     SUB_COM.SUBSYS_ID = PROJ_SUB.SUBSY_ID
  5    AND     PROJ_SUB. PROJ_NO = PROJECT. PROJ_NO
  6    AND     PROJ_NAME = <user-supplied project name>
  7    AND     SUB_PRE = <user-supplied subsystem prefix>
  8    AND     COM_NAME = <user-supplied component name>;
```

Example 3

This example uses a predefined view as an alternative to the method presented in example 2 to get the same data (i.e., the date on which a component was entered into the controlled library). The access path for using the view V_PROJ_COM to retrieve this data item is as follows:



In this example, view V_PROJ_COM replaces tables PROJECT, PROJ_SUB, and SUB_COM used in the previous example. The view already joins these tables. The result is field CREATE_DATE of the COM_SOURCE table.

SQL statement

```
SQL>   SELECT  CREATE_DATE
  2    FROM    V_PROJ_COM, COM_SOURCE
  3    WHERE   V_PROJ_COM.COM_NO = COM_SOURCE.COM_NO
  4    AND     COM_NAME = <user-supplied component name>
  5    AND     SUB_PRE = <user-supplied subsystem prefix>
  6    AND     PROJ_NAME = <user-supplied project name>;
```

The SQL statements in these examples are included for completeness. For a more detailed introduction to formulating SQL queries, see Section 5.3.

Table 4-4. SEL Database Access Paths (1 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P63, D82	CHANGE	CHANGE_NO	PROJECT NAME	[PROJ_NAME] → V_PROJ_COM ↓ [COM_NO] CHANGE_COM ↓ [CHANGE_NO] CHANGE → [CHANGE_NO]
P76, D67	CHANGE	CH_TYPE	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] → CHANGE ↓ [CH_TYPE]*CODED FIELD
P73, D64	CHANGE	DATA_COMP	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] → CHANGE ↓ [DATE_COMP]
P72, D63	CHANGE	DATE_DETER	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] → CHANGE ↓ [DATE_DETER]
P69, D76	CHANGE	EFF_ADA	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] → CHANGE ↓ [EFF_ADA]
P67, D66	CHANGE	EFF_COM_CH	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] → CHANGE ↓ [EFF_COM_CH]*CODED FIELD]
P66, D65	CHANGE	EFF_ISO_CH	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] → CHANGE ↓ [EFF_ISO_CH]*CODED FIELD]

Table 4-4. SEL Database Access Paths (2 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P68, D68	CHANGE	EFF_ONE	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CHANGE ↓ [EFF_ONE]
P70, D69	CHANGE	EFF_OTHER	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CHANGE ↓ [EFF_OTHER]
P71, D70	CHANGE	EFF_PARPA	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CHANGE ↓ [EFF_PARPA]
P74	CHANGE	NUM_COM_CH	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CHANGE ↓ [NUM_COM_CH]
P75	CHANGE	NUM_COM_EX	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CHANGE ↓ [NUM_COM_EX]
P65, D2	CHANGE	SUB_DATE	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CHANGE ↓ [SUB_DATE]
P85, D77	CH_ADAFEAT	ADA_FEATURE	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CH_ADAFEAT ↓ [ADA_FEATURE]*CODED FIELD

Table 4-4. SEL Database Access Paths (3 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P86, D80	CH_ERR_ARES	ERR_ARES	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CH_ERR_ARES ↓ [ERR_ARES]*CODED FIELD
P83, D79	CH_ERR_GEN	ERR_ACAUSE	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CH_ERR_GEN ↓ [ERR_ACAUSE]*CODED FIELD
P82, D78	CH_ERR_GEN	ERR_ADOC	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CH_ERR_GEN ↓ [ERR_ADOC]
P78, D72	CH_ERR_GEN	ERR_CLASS	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CH_ERR_GEN ↓ [ERR_CLASS]*CODED FIELD
P79, D74	CH_ERR_GEN	ERR_COMIS	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CH_ERR_GEN ↓ [ERR_COMIS]
P80, D73	CH_ERR_GEN	ERR_OMIS	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CH_ERR_GEN ↓ [ERR_OMIS]
P77, D71	CH_ERR_GEN	ERR_SOURCE	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO]→ CH_ERR_GEN ↓ [ERR_SOURCE]*CODED FIELD

Table 4-4. SEL Database Access Paths (4 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P81, D75	CH_ERR_GEN	ERR_TYPO	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] → CH_ERR_GEN ↓ [ERR_TYPO]
P87, D81	CH_ERR_TOOLS	ERR_TOOLS	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] → CH_ERR_TOOLS ↓ [ERR_TOOLS]*CODED FIELD
M4	COMPUTER	CPU_NAME	NONE	COMPUTER → [CPU_NAME]
M5	COMPUTER	C_FULL_NAME	NONE	[CPU_NAME] → COMPUTER → [C_FULL_NAME]
P59, D58	COM_PURPOSE	PURPOSE	PROJECT NAME, SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] [COM_NAME] → SUB_COM ↓ [COM_NO] COM_PURPOSE ↓ [PURPOSE]*CODED FIELD
P58, D57	COM_SOURCE	COM_TYPE	PROJECT NAME, SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] [COM_NAME] → SUB_COM ↓ [COM_NO] COM_SOURCE ↓ [COM_TYPE]*CODED FIELD

Table 4-4. SEL Database Access Paths (5 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P53, D54	COM_ SOURCE	CREATE_ DATE	PROJECT NAME, SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] [COM_NAME] → SUB_COM ↓ [COM_NO] COM_SOURCE ↓ [CREATE_DATE]
P57, D55	COM_ SOURCE	DIFFICULTY	PROJECT NAME, SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] [COM_NAME] → SUB_COM ↓ [COM_NO] COM_SOURCE ↓ [DIFFICULTY]
D59	COM_ SOURCE	FORM_NO	PROJECT NAME, SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] [COM_NAME] → SUB_COM ↓ [COM_NO] COM_SOURCE ↓ [FORM_NO]

Table 4-4. SEL Database Access Paths (6 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P56, D56	COM_ SOURCE	ORI_TYPE	PROJECT NAME, SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] [COM_NAME] → SUB_COM ↓ [COM_NO] COM_SOURCE ↓ [ORI_TYPE]*CODED FIELD
P54, D2	COM_ SOURCE	SUB_DATE	PROJECT NAME, SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] [COM_NAME] → SUB_COM ↓ [COM_NO] COM_SOURCE ↓ [SUB_DATE]
P156	COM_ STAT	C_C_LINE	PROJECT NAME AND COM-PONENT NAME	[PROJ_NAME] → V_PROJ_COM ← [COM_NAME] ↓ [COM_NO] COM_STAT ↓ C_C_LINE
P154	COM_ STAT	C_EXE_S	PROJECT NAME AND COM-PONENT NAME	[PROJ_NAME] → V_PROJ_COM ← [COM_NAME] ↓ [COM_NO] COM_STAT ↓ [C_EXE_S]

Table 4-4. SEL Database Access Paths (7 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P155	COM_ STAT	C_LINE	PROJECT NAME AND COM-PONENT NAME	[PROJ_NAME] → V_PROJ_COM ← [COM_NAME] ↓ [COM_NO] COM_STAT ↓ [C_LINE]
P221	COM_ STAT	C_STMT	PROJECT NAME AND COM-PONENT NAME	[PROJ_NAME] → V_PROJ_COM ← [COM_NAME] ↓ [COM_NO] COM_STAT ↓ [C_STMT]
P222	COM_ STAT	FINAL_ ORIGIN_CAT	PROJECT NAME AND COM-PONENT NAME	[PROJ_NAME] → V_PROJ_COM ← [COM_NAME] ↓ [COM_NO] COM_STAT ↓ [FINAL_ORIGIN_CAT]
P196, D181, P198, D183, P200–P202, D185–D186, P204–P206, D189–D190, P208, D193, P210, D195, P212, D197	DSF_ MEASURE	MEASURE_ VALUE	PROJECT NAME AND MEASURE-MENT CODE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_DSF ↓ [D_ID] [MEASURE_CODE] → DSF_MEASURE ↓ [MEASURE_VALUE] WHERE MEASURE_CODE FOR P196, D181 = MODDESIGN MEASURE_CODE FOR P198, D183 = MODCODE MEASURE_CODE FOR P200, D185 = SYSTSTONE MEASURE_CODE FOR P201, D186 = SYSTSTPASS MEASURE_CODE FOR P202 = SYSTSTRUN MEASURE_CODE FOR P204, D189 = ACCTSTONE MEASURE_CODE FOR P205, D190 = ACCTSTPASS MEASURE_CODE FOR P206 = ACCTSTRUN MEASURE_CODE FOR P208, D193 = DISCRES MEASURE_CODE FOR P210, D195 = SPECMODIMP MEASURE_CODE FOR P212, D19 = QUESTANS

Table 4-4. SEL Database Access Paths (8 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P195, D180, P197, D182, P199, D184, P203, D188, P207, D192, P209, D194, P211, D196	DSF_ TARGET	TARGET_ VALUE	PROJECT NAME AND TARGET_CODE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_DSF ↓ [D_ID] [TARGET_CODE] → DSF_TARGET ↓ [TARGET_VALUE] WHERE TARGET_CODE FOR P195, D180 = TOTDESIGN TARGET_CODE FOR P197, D182 = TOTCODE TARGET_CODE FOR P199, D184 = TOTSYSTST TARGET_CODE FOR P203, D188 = TOTACCTST TARGET_CODE FOR P207, D192 = TOTDISCREP TARGET_CODE FOR P209, D194 = SPECMODREC TARGET_CODE FOR P211, D196 = QUESTSUB
P25–P34, D23–D32, P157–P166, D199–D208	EFF_ACT	ACT_HR (FROM PRF OR CLPRF)	PROJECT NAME, PROGRAMMER NAME, WEEK ENDING DATE, SUBSYSTEM PREFIX (OPTIONAL), AND ACTIVITY	[PROJ_NAME] → PROJECT [FORM_NAME] ↓ [PROJ_NO] PERSONNEL ↓ [PROG_ID] → EFF_PROJ ← [SUB_DATE] [P_ID] [P_ID] EFF_SUB ← [SUB_PRE] [ACTIVITY] → EFF_ACT ← [PS_ID] ↓ [ACT_HR] WHERE (FOR PRF) ACTIVITY FOR P25, D2 = PREDES ACTIVITY FOR P26, D24 = CREDES ACTIVITY FOR P27, D25 = RDREVDES ACTIVITY FOR P28, D26 = WRCODE ACTIVITY FOR P29, D27 = RDREVCOD ACTIVITY FOR P30, D28 = TSTCODUN ACTIVITY FOR P31, D29 = DEBUG ACTIVITY FOR P32, D30 = INTTEST ACTIVITY FOR P33, D31 = ACCTEST ACTIVITY FOR P34, D32 = OTHER

Table 4-4. SEL Database Access Paths (9 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P25–P34, D23–D32, P157–P166, D199–D208 (Cont'd)	EFF_ACT	ACT_HR (FROM PRF OR CLPRF)	PROJECT NAME, PROGRAMMER NAME, WEEK ENDING DATE, SUBSYSTEM PREFIX (OPTIONAL), AND ACTIVITY	(FOR CLPRF) ACTIVITY FOR P157, D199 = CLPREDES ACTIVITY FOR P158, D200 = CLPRETEST ACTIVITY FOR P159, D201 = CLCREDES ACTIVITY FOR P160, D202 = CLVEREVDES ACTIVITY FOR P161, D203 = CLWRCODE ACTIVITY FOR P162, D204 = CLRDREVCOD ACTIVITY FOR P163, D205 = CLINDTEST ACTIVITY FOR P164, D206 = CLRESPSFR ACTIVITY FOR P165, D20 = CLACCTEST ACTIVITY FOR P168, D208 = CLOTHER
P157–P166, D199–D208	EFF_ACT	ACT_HR (FROM CLPRF, MAPPED TO PRF ACTIVITIES)	CLEANROOM PROJECT NAME, PROGRAMMER NAME, AND WEEK ENDING DATE *CLEANROOM ACTIVITIES ARE CONVERTED TO STANDARD ACTIVITIES BY USING V_CLEANROOM ACT	[CLEANROOM PROJ_NAME] → PROJECT [FORM_NAME] → PERSONNEL → [PROG_ID] → EFF_PROJ ← [PROJ_NO], [SUB_DATE] EFF_PROJ → [P_ID] [ACTIVITY] → V_CLEANROOM_ACT → [ACT_HR] WHERE ACTIVITY FOR P25, D23 = PREDES ACTIVITY FOR P26, D24 = CREDES ACTIVITY FOR P27, D25 = RDREVCOD ACTIVITY FOR P28, D26 = WRCODE ACTIVITY FOR P29, D27 = RDREVDES ACTIVITY FOR P31, D29 = DEBUG ACTIVITY FOR P32, D30 = INTTEST ACTIVITY FOR P33, D31 = ACCTEST ACTIVITY FOR P34, D32 = OTHER

Table 4-4. SEL Database Access Paths (10 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P39, P40, P42, P43, D44, D45, D47, D48	EFF_ACT	ACT_HR (FROM SPF)	PROJECT NAME, PROGRAMMER NAME, AND WEEK ENDING DATE	[PROJ_NAME] → PROJECT [FORM_NAME] [PROJ_NO] ↓ ↓ PERSONNEL ↓ ↓ [PROG_ID] → EFF_PROJ ← [SUB_DATE] ↓ [P_ID] = [EFF_ID] [ACTIVITY] → EFF_ACT ↓ [ACT_HR] WHERE FORM_NAME FOR P39, D44 = TECHPUBS FORM_NAME FOR P40, D45 = SECRTARY FORM_NAME FOR P42, D47 = PROGMGMT FORM_NAME FOR P43, D48 = OTHSUPP AND ACTIVITY FOR P39, D44, P40, D45, P42, D47, P43, D48 = SUPPORT
D37, D49, D210	EFF_ FORM	FORM_NO	PROJECT NAME AND FORM TYPE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] EFF_PROJ ↓ [P_ID] [FORM_TYPE] → EFF_FORM ↓ [FORM_NO] WHERE FORM_TYPE FOR D37 = PRF FORM_TYPE FOR D49 = SPF FORM_TYPE FOR D210 = CLPRF
P23, D22	EFF_ PROJ	SUB_DATE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] EFF_PROJ ↓ [SUB_DATE]

Table 4-4. SEL Database Access Paths (11 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P172–P177, D155–D160	MAINT_ ACT_HRS	ACT_HR	PROJECT NAME, PROGRAMMER NAME, WEEK ENDING DATE, AND MAINTE-NANCE ACTIVITY	[PROJ_NAME] → PROJECT [PROJ_NO] [FORM_NAME] ↓ PERSONNEL ↓ [PROG_ID] → MAINT_PROJ ← [SUB_DATE] [MAINT_NO] ↓ [MAINT_ACT] → MAINT_ACT_HRS ↓ [ACT_HR] WHERE MAINT_ACT FOR P172, D155 = ISOLATION MAINT_ACT FOR P173, D156 = REDESIGN MAINT_ACT FOR P174, D157 = IMPLEMENT MAINT_ACT FOR P175, D158 = UNSYSTEST MAINT_ACT FOR D176, D159 = ACCBENTEST MAINT_ACT FOR P177, D160 = OTHER
P180, D164	MAINT_ CHANGE	CH_CAUSE	MAINTENANCE CHANGE NUM-BER; SEE D178 FOR THE AC-CESS PATH THAT FINDS A PARTIC-ULAR MAINTE-NANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [CH_CAUSE]*CODED FIELD
P184, D168	MAINT_ CHANGE	CH_CLASS	MAINTENANCE CHANGE NUM-BER; SEE D178 FOR THE AC-CESS PATH THAT FINDS A PARTIC-ULAR MAINTE-NANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [CH_CLASS]*CODED FIELD
P188, D172	MAINT_ CHANGE	COMP_ADD	MAINTENANCE CHANGE NUM-BER; SEE D178 FOR THE AC-CESS PATH THAT FINDS A PARTIC-ULAR MAINTE-NANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [COMP_ADD]

Table 4-4. SEL Database Access Paths (12 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P191, D175	MAINT_ CHANGE	COMP_ADD_ NEW	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [COMP_ADD_NEW]
P193, D177	MAINT_ CHANGE	COMP_ADD_ REMOD	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [COMP_ADD_REMOD]
P192, D176	MAINT_ CHANGE	COMP_ADD_ REUSE	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [COMP_ADD_REUSE]
P189, D173	MAINT_ CHANGE	COMP_CH	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [COMP_CH]
P190, D174	MAINT_ CHANGE	COMP_DEL	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [COMP_DEL]
P185, D169	MAINT_ CHANGE	EST_LOC_ ADD	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [EST_LOC_ADD]

Table 4-4. SEL Database Access Paths (13 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P186, D170	MAINT_ CHANGE	EST_LOC_ CH	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [EST_LOC_CH]
P187, D171	MAINT_ CHANGE	EST_LOC_ DEL	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [EST_LOC_DEL]
D178	MAINT_ CHANGE	MAINT_CH_ NO	PROJECT NAME, PROGRAMMER NAME, AND SUBMISSION DATE	[PROJ_NAME] → PROJECT [PROJ_NO] [FORM_NAME] ↓ PERSONNEL ↓ [PROG_ID] → MAINT_CHANGE ↓ [MAINT_CH_NO]
P179, D163	MAINT_ CHANGE	MAINT_CH_ TYPE	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [MAINT_CH_TYPE]*CODED FIELD
P182, D166	MAINT_ CHANGE	MAINT_COM_ CH	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [MAINT_COM_CH]*CODED FIELD

Table 4-4. SEL Database Access Paths (14 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P181, D165	MAINT_ CHANGE	MAINT_ISO_ CH	MAINTENANCE CHANGE NUMBER; SEE D178 FOR THE ACCESS PATH THAT FINDS A PARTICULAR MAINTENANCE CHANGE NUMBER	[MAINT_CH_NO] → MAINT_CHANGE ↓ [MAINT_ISO_CH]*CODED FIELD
P183, D167	MAINT_ CH_ OBJECTS	CH_OBJECT	MAINT CHANGE NUMBER	[MAINT_CH_NO] ↓ MAINT_CH_OBJECTS ↓ [CH_OBJECT]*CODED FIELD]
P168–P171, D151–D154	MAINT_ CLASS_ HRS	CLASS_HR	PROJECT NAME, PROGRAMMER NAME, AND WEEK ENDING DATE	[PROJ_NAME] → PROJECT [FORM_NAME] [PROJ_NO] ↓ PERSONNEL ↓ [PROG_ID] → MAINT_PROJ ← [SUB_DATE] [MAINT_ID] ↓ [MAINT_CLASS] → MAINT_CLASS_HRS ↓ [CLASS_HR] WHERE MAINT_CLASS FOR P168, D151 = CORRECTION MAINT_CLASS FOR P169, D152 = ENHANCEMNT MAINT_CLASS FOR P170, D153 = ADAPTATION MAINT_CLASS FOR P171, D154 = OTHER
P23, D22	MAINT_ PROJ	SUB_DATE	PROJECT NAME	[PROJ_NAME] → PROJECT FORM_NAME [PROJ_NO] ↓ PERSONNEL ↓ [PROG_ID] → MAINT_PROJ ↓ [SUB_DATE]
M1	PERSONNEL	FORM_NAME	NONE	PERSONNEL → [FORM_NAME]
M3	PERSONNEL	DATE_ENTRY	PROGRAMMER NAME	[FORM_NAME]→ PERSONNEL→ [DATE_ENTRY]

Table 4-4. SEL Database Access Paths (15 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P24, D21	PERSON-NEL	FORM_NAME (FROM COF)	PROJECT NAME, SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] [COM_NAME] → SUB_COM ↓ [COM_NO] COM_SOURCE ↓ [PROG_ID] PERSONNEL ↓ [FORM_NAME]
P24, D21	PERSON-NEL	FORM_NAME (FROM CRF)	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] → CHANGE ↓ [PROG_ID] PERSONNEL ↓ [FORM_NAME]
P24, D21	PERSON-NEL	FORM_NAME (FROM DSF)	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_DSF ↓ [PROG_ID] PERSONNEL ↓ [FORM_NAME]
P24, D21	PERSON-NEL	FORM_NAME (FROM MCRF)	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] MAINT_CHANGE ↓ [PROG_ID] PERSONNEL ↓ [FORM_NAME]

Table 4-4. SEL Database Access Paths (16 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P24, D21	PERSON-NEL	FORM_NAME (FROM PRF OR CLPRF)	PROJECT NAME AND FORM TYPE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [FORM_TYPE] → EFF_PROJ ↓ [PROG_ID] PERSONNEL ↓ [FORM_NAME] WHERE FORM_TYPE = PRF OR CLPRF
P24, D21	PERSON-NEL	FORM_NAME (FROM SPF)	PROJECT NAME AND FORM TYPE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [FORM_TYPE] → EFF_PROJ ↓ [PROG_ID] PERSONNEL ↓ [FORM_NAME] WHERE FORM_TYPE = SPF NOTE: FORM_NAME = LIBARIAN, OTHSUPP, PROGMGMT, SECRTARY, TECHPUBS
P24, D21	PERSON-NEL	FORM_NAME (FROM WMEF)	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] MAINT_PROJ ↓ [PROG_ID] PERSONNEL ↓ [FORM_NAME]
M2	PERSON-NEL	FULL_NAME	PROGRAMMER NAME	[FORM_NAME] → PERSONNEL → [FULL_NAME]
P3	PROJECT	ACTIVE_ STATUS	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [ACTIVE_STATUS]*CODED FIELD
P1, D1	PROJECT	PROJ_NAME	NONE	PROJECT → [PROJ_NAME]
P2, D163	PROJECT	PROJ_TYPE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_TYPE]*CODED FIELD

Table 4-4. SEL Database Access Paths (17 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P134, D38	PROJ_ CPU_ STAT	CPU_NAME	PROJECT NAME	[PROJ_NAME]→ PROJECT ↓ [PROJ_NO] PROJ_CPU_STAT ↓ [CPU_NAME]
P135, D94	PROJ_ CPU_ STAT	TOTAL_HRS	PROJECT NAME	[PROJ_NAME]→ PROJECT ↓ [PROJ_NO] PROJ_CPU_STAT ↓ [TOTAL_HRS]
P136, D95	PROJ_ CPU_ STAT	T_RUN	PROJECT NAME	[PROJ_NAME]→ PROJECT ↓ [PROJ_NO] PROJ_CPU_STAT ↓ [T_RUN]
P23, D22	PROJ_ DSF	SUB_DATE	PROJECT NAME	[PROJ_NAME]→ PROJECT ↓ [PROJ_NO] PROJ_DSF ↓ [SUB_DATE]
P21, D12	PROJ_ EST	MAN_HR	PROJECT NAME AND SUBMIS- SION DATE OF DESIRED SET OF ESTIMATES	[PROJ_NAME]→ PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST ↓ [MAN_HR]
P20, D11	PROJ_ EST	PRO_HR	PROJECT NAME AND SUBMIS- SION DATE OF DESIRED SET OF ESTIMATES	[PROJ_NAME]→ PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST ↓ [PRO_HR]
P22, D13	PROJ_ EST	SER_HR	PROJECT NAME AND SUBMIS- SION DATE OF DESIRED SET OF ESTIMATES	[PROJ_NAME]→ PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST ↓ [SER_HR]

Table 4-4. SEL Database Access Paths (18 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P13, D2	PROJ_ EST	SUB_DATE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_EST ↓ [SUB_DATE]
P15, D15	PROJ_ EST	T_COM	PROJECT NAME AND SUBMIS-SION DATE OF DESIRED SET OF ESTIMATES	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST ↓ [T_COM]
P16, D16	PROJ_ EST	T_LINE	PROJECT NAME AND SUBMIS-SION DATE OF DESIRED SET OF ESTIMATES	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST ↓ [T_LINE]
P18, D18	PROJ_ EST	T_MOD_LINE	PROJECT NAME AND SUBMIS-SION DATE OF DESIRED SET OF ESTIMATES	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST ↓ [T_MOD_LINE]
P19, D17	PROJ_ EST	T_NEW_LINE	PROJECT NAME AND SUBMIS-SION DATE OF DESIRED SET OF ESTIMATES	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST ↓ [T_NEW_LINE]
P17, D19	PROJ_ EST	T_OLD_LINE	PROJECT NAME AND SUBMIS-SION DATE OF DESIRED SET OF ESTIMATES	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST ↓ [T_OLD_LINE]
P14, D14	PROJ_ EST	T_SYS	PROJECT NAME AND SUBMIS-SION DATE OF DESIRED SET OF ESTIMATES	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST ↓ [T_SYS]

Table 4-4. SEL Database Access Paths (19 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P10, D91	PROJ_ EST_ PHASE	END_DATE	PROJECT NAME AND SUBMIS-SION DATE OF PEF OR PCSF	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST_PHASE ↓ MAX [END_DATE]
P6–P11, D3–D8, P125–P131, D84–D90	PROJ_ EST_ PHASE	START_DATE	PROJECT NAME AND SUBMIS-SION DATE OF PEF OR PCSF	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_EST_PHASE ↓ MIN [START_DATE]
P6–P11,	PROJ_ EST_ PHASE	START_DATE, END_DATE	PROJECT NAME, PHASE CODE, AND SUBMIS-SION DATE OF PEF OR PCSF	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE]→ PROJ_EST_PHASE ← [PHASE_ CO] ↓ [START_DATE], [END_DATE] WHERE PHASE_CO FOR P6, D3, P125, D84 = REQNT PHASE_CO FOR P7, D4, P126, D85 = DESGN PHASE_CO FOR P8, D5, P127, D86 = CODET PHASE_CO FOR P9, D6, P128, D87 = SYSTE PHASE_CO FOR P10, D7, P129, D88 = ACCTE PHASE_CO FOR P11, D8, P130, D89 = CLEAN PHASE_CO FOR P131, D90 = MAINT
P5, P13, P124, D2	PROJ_ EST_ PHASE	SUB_DATE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_EST_PHASE ↓ [SUB_DATE]
D20, D49, D113, D150	PROJ_ FORM	FORM_NO	PROJECT NAME, AND FORM TYPE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [FORM_TYPE] → PROJ_FORM ↓ [FORM_NO] WHERE FORM_TYPE FOR D150 = SEF FORM_TYPE FOR D20 = PEF FORM_TYPE FOR D49 = SPF FORM_TYPE FOR D113 = PCSF

Table 4-4. SEL Database Access Paths (20 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P62, D42	PROJ_ GRH	GR_CH	PROJECT NAME, AND WEEK END-ING DATE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_GRH ↓ [GR_CH]
P60, D43	PROJ_ GRH	GR_LINE	PROJECT NAME AND WEEK END-ING DATE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_GRH ↓ [GR_LINE]
P61, D41	PROJ_ GRH	GR_MOD	PROJECT NAME AND WEEK END-ING DATE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_GRH ↓ [GR_MOD]
P4, D62	PROJ_ MES-SAGES	MESSAGES	PROJECT NAME AND NOTE TYPE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [NOTE_TYPE] → PROJ_NOTES ↓ [S_ID] PROJ_MESSAGES ↓ [MESSAGES] WHERE NOTE_TYPE = CLOSEOUT, COMPACCTS, COMPSYS, CONTACTS, CONTRLLIB, DATAAVAIL, FORMSCOL, GENMESS, GHTOOL, LANGUAGES, PROJNAME, OR TASKNO
P4, D61	PROJ_ NOTES	NOTE_TYPE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_NOTES ↓ [NOTE_TYPE]*CODED FIELD

Table 4-4. SEL Database Access Paths (21 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P45, D39	PROJ_ PROD	RES_HR	PROJECT NAME COMPUTER NAME, AND WEEK ENDING DATE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_PROD ← [RES_NAME] ↓ [RES_HR]
P44, D38	PROJ_ PROD	RES_NAME	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_PROD ↓ [RES_NAME]
P46, D40	PROJ_ PROD	RES_RUN	PROJECT NAME, COMPUTER NAME, AND WEEK ENDING DATE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_DATE] → PROJ_PROD ← [RES_NAME] ↓ [RES_RUN]

Table 4-4. SEL Database Access Paths (22 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P88–P107, D114–D133, P109–P123, D135–D149	PROJ_ SEF	EVALUATE	PROJECT NAME AND MEASURE-MENT TYPE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [MEAS_TYPE] → PROJ_SEF ↓ [EVALUATE] WHERE MEAS_TYPE FOR P88, D114 = PM01 MEAS_TYPE FOR P89, D115 = PM02 MEAS_TYPE FOR P90, D116 = PM03 MEAS_TYPE FOR P91, D117 = PM04 MEAS_TYPE FOR P92, D118 = PM05 MEAS_TYPE FOR P93, D119 = PM06 MEAS_TYPE FOR P94, D120 = ST07 MEAS_TYPE FOR P95, D121 = ST08 MEAS_TYPE FOR P96, D122 = ST09 MEAS_TYPE FOR P97, D123 = ST10 MEAS_TYPE FOR P98, D124 = TM11 MEAS_TYPE FOR P99, D125 = TM12 MEAS_TYPE FOR P100, D126 = TM13 MEAS_TYPE FOR P101, D127 = TM14 MEAS_TYPE FOR P102, D128 = TM15 MEAS_TYPE FOR P103, D129 = PC16 MEAS_TYPE FOR P104, D130 = PC17 MEAS_TYPE FOR P105, D131 = PC18 MEAS_TYPE FOR P106, D132 = PC19 MEAS_TYPE FOR P107, D133 = PC20 MEAS_TYPE FOR P109, D135 = PC22 MEAS_TYPE FOR P110, D136 = PC23 MEAS_TYPE FOR P111, D137 = PC24 MEAS_TYPE FOR P112, D138 = EN25 MEAS_TYPE FOR P113, D139 = EN26 MEAS_TYPE FOR P114, D140 = EN27 MEAS_TYPE FOR P115, D141 = EN28 MEAS_TYPE FOR P116, D142 = EN29 MEAS_TYPE FOR P117, D143 = EN30 MEAS_TYPE FOR P118, D144 = PT31 MEAS_TYPE FOR P119, D145 = PT32 MEAS_TYPE FOR P120, D146 = PT33 MEAS_TYPE FOR P121, D147 = PT34 MEAS_TYPE FOR P122, D148 = PT35 MEAS_TYPE FOR P123, D149 = PT36
P108, D134	PROJ_ SEF_SEC	SECOND_L	PROJECT NAME AND MEASURE-MENT TYPE	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [MEAS_TYPE] → PROJ_SEF_SEC ↓ [SECOND_L]*CODED FIELD NOTE: MEAS_TYPE = PC21

Table 4-4. SEL Database Access Paths (23 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P133, D93	PROJ_ STAT	SER_HR	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [SER_HR]
P132, D92	PROJ_ STAT	TECH_MAN_ HR	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [TECH_MAN_HR]
P139, D98	PROJ_ STAT	T_CH	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_CH]
P138, D97	PROJ_ STAT	T_COM	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_COM]
P145, D104	PROJ_ STAT	T_COMMENT	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_COMMENT]
P140, D99	PROJ_ STAT	T_DOC	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_DOC]
P146, D105	PROJ_ STAT	T_EXE_MOD	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_EXE_MOD]

Table 4-4. SEL Database Access Paths (24 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P150, D109	PROJ_ STAT	T_EXE_STAT	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_EXE_STAT]
P213, D211	PROJ_ STAT	T_EXTMO_ LINE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_EXTMO_LINE]
P214, D212	PROJ_ STAT	T_EXTMO_ MOD	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_EXTMO_MOD]
P215, D213	PROJ_ STAT	T_EXTMO_ STAT	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_EXTMO_STAT]
P219, D217	PROJ_ STAT	T_EXTMO_ STMTS	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_EXTMO_STMTS]
P141, D100	PROJ_ STAT	T_LINE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_LINE]
P143, D102	PROJ_ STAT	T_MOD_LINE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_MOD_LINE]

Table 4-4. SEL Database Access Paths (25 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P148, D107	PROJ_ STAT	T_MOD_MOD	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_MOD_MOD]
P152, D111	PROJ_ STAT	T_MOD_STAT	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_MOD_STAT]
P218, D216	PROJ_ STAT	T_MOD_ STMTS	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_MOD_STMTS]
P142, D101	PROJ_ STAT	T_NEW_LINE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_NEW_LINE]
P147, D106	PROJ_ STAT	T_NEW_MOD	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_NEW_MOD]
P151, D110	PROJ_ STAT	T_NEW_STAT	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_NEW_STAT]
P217, D215	PROJ_ STAT	T_NEW_ STMTS	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_NEW_STMTS]

Table 4-4. SEL Database Access Paths (26 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P144, D103	PROJ_ STAT	T_OLD_LINE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_OLD_LINE]
P149, D108	PROJ_ STAT	T_OLD_MOD	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_OLD_MOD]
P153, D112	PROJ_ STAT	T_OLD_STAT	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_OLD_STAT]
P220, D218	PROJ_ STAT	T_OLD_ STMTS	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_OLD_STMTS]
P216, D214	PROJ_ STAT	T_STMTS	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_STMTS]
P137, D96	PROJ_ STAT	T_SYS	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_SYS]
P47, P84, D50	PROJ_ SUB	SUB_PRE	PROJECT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] PROJ_SUB ↓ [SUB_PRE]

Table 4-4. SEL Database Access Paths (27 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P50, D2	PROJ_ SUB	SUB_DATE	PROJECT NAME AND SUBSYS-TEM PREFIX	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUB_DATE]
P35–P38, D33–D36, P167, D209	SPECIAL_ ACT	ACT_HR	PROJECT NAME PROGRAMMER NAME, WEEK ENDING DATE, AND SPECIAL ACTIVITY	[PROJ_NAME] → PROJECT [FORM_NAME] [PROJ_NO] ↓ PERSONNEL ↓ [PROG_ID] → EFF_PROJ ← [SUB_DATE] [P_ID] = [EFF_ID] ↓ [ACTIVITY] → SPECIAL_ACT ↓ [ACT_HR] WHERE (FOR PRF) SP_ACTIVITY FOR P35, D33 = REWORK SP_ACTIVITY FOR P36, D34 = ENHANCE SP_ACTIVITY FOR P37, D35 = DOCUMENT SP_ACTIVITY FOR P38, D36 = REUSE (FOR CLPRF) SP_ACTIVITY FOR P167, D209 = CLMETHOD
P49, D52	SUBSYS-TEM	FUNCTION	PROJECT NAME AND SUBSYS-TEM PREFIX	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] SUBSYSTEM ↓ [FUNCTION]*CODED FIELD
P48, D51	SUBSYS-TEM	NAME	PROJECT NAME AND SUBSYS-TEM PREFIX	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] SUBSYSTEM ↓ [NAME]

Table 4-4. SEL Database Access Paths (28 of 28)

Ref. ID	Target Table	Target Column	Access Information	Access Path
P51, D53	SUB_COM	COM_NAME	PROJECT NAME AND SUBSYS-TEM PREFIX	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] SUB_COM ↓ [COM_NAME]
P52, D2	SUB_COM	COM_DATE	PROJECT NAME SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] → PROJECT ↓ [PROJ_NO] [SUB_PRE] → PROJ_SUB ↓ [SUBSY_ID] [COM_NAME] → SUB_COM ↓ [COM_DATE]
P84, D53	V_PROJ_ COM	COM_NAME	PROJECT NAME	CHANGE_COM ↓ [COM_NO] [PROJ_NAME] → V_PROJ_COM ↓ [COM_NAME]

SECTION 5—ACCESSING THE SEL DATABASE

The database table definitions, relationships, and access paths presented in Section 4 provide a guide to finding a particular software engineering data item in the database. This section discusses how to actually access a data item once its location in the schema has been identified.

Section 5.1 discusses how a user initially obtains access to the SEL database. Section 5.2 provides an introduction to the DAMSEL user interface (UI) subsystem: menus that allow users to view data, enter data, generate reports, and perform various database support functions. Section 5.3 presents an introduction to ad hoc database queries via SQL*Plus, which is provided by ORACLE. This introduction covers the basics of how to formulate a SQL query and provides several illustrative examples. Section 5.4 presents an introduction to the query library. This introduction covers the help system, searching the library, and executing and spooling queries.

5.1 DATABASE ACCESS REQUIREMENTS

To access the SEL database through SQL*Plus, a user must have a user ID and password for the STL VAX 11/780 and an ORACLE user ID and password on the VAX. To access the SEL database through DAMSEL, a user must have these IDs and passwords, plus have their ORACLE user ID enrolled as a DAMSEL user. All of these can be obtained by contacting either STL systems personnel or the SEL DBA at CSC. In DAMSEL, user classes are defined to give different types of users appropriate levels of database access. The user class determines the access privileges a user has with respect to individual database tables and the functions that may be performed in DAMSEL. The following user classes have been defined:

- General user—Users requiring read-only access to the database, such as researchers and managers
- Librarian—SEL data entry personnel
- QA—SEL quality assurance personnel
- Maintenance—SEL database maintenance programmers
- DBA—SEL database administrator

Once a user obtains the appropriate accounts and privileges and logs onto the STL VAX, the user must execute the following command procedure to create all of the logicals and symbols required to access the ORACLE RDBMS and the DAMSEL system:

```
$ @STL DISK1:[TOOLS]SELINIT
```

To avoid having to type this command each time the user logs on the VAX to access the database, it is recommended that the command be included in the user's LOGIN.COM file. Then it will be executed automatically whenever the user logs onto the VAX.

5.2 DAMSEL

DAMSEL provides a convenient way for all classes of users to access the SEL data. This menu-driven user interface has five major options at the top level:

- **Form function option**—This option permits users to view, insert, update, delete, or quality assure SEL data interactively, one SEL form at a time. The screens for performing these operations display data in a manner that resembles the data collection forms presented in Section 3.

- **Report function option**—This selection provides a method for users to view large amounts of data on single projects, or on multiple projects, within a single report. Reports are available for viewing data that are not project-specific or related to SEL forms. Users select a sequence of reports and options (a script) from the report menus and submit the script to be executed. They may also save frequently used report scripts for future execution. Reports can be submitted interactively or as batch jobs. The results may be printed or routed to files for terminal display and/or future printing.

- **Query support function option**—This selection provides a set of ad hoc SQL queries that would likely be used by general users, such as researchers and managers. (This option is currently not available.)

- **DBA function option**—This selection provides data entry screens for the SEL DBA to enter or modify projects, personnel information, and computer information and to perform various database verification tasks.

- **General database support function option**—This selection provides to SEL database support personnel the capability to generate distribution tapes.

Users, depending on their assigned user class, may have access to one or more of these functions. The menu system has built-in security features to verify that each user has the access privilege to the functions that he or she is attempting to perform. The message "You do not have access to this option" will appear on the screen if the user tries to perform a function that is not in his/her operational domain. Each user class has different access privileges in the menu system. These are defined as follows:

- **General user**—This class of user can access all the SEL form function viewing screens, all the report function screens, and all the query support function screens.

- **Librarian**—This class of user can access all the SEL form function viewing, insert, update, and delete screens; all the report function screens; and the general database support function screens.

- **QA**—This class of user can access all the SEL form function viewing and quality assurance screens, plus all the report function screens.
- **Maintenance**—This class of user can access all the SEL form function viewing screens, all the report function screens, all the query support function screens, and the general support function screens.
- **DBA**—This class of user can access all the SEL form function viewing screens, all the report function screens, all the query support function screens, all the DBA function screens, and all the general support function screens.

After the database access requirements, described in Section 5.1, are satisfied, the user can access DAMSEL as follows:

- Log on to the VAX using his/her VAX user ID and password.
- At the '$' prompt, type DAMSEL.
- Enter his/her ORACLE user ID and password at the prompts on the DAMSEL login screen.
- Select menu options.
- Terminate the DAMSEL session via the <Exit/Cancel> key.

Reference 3 presents a more detailed discussion on using the DAMSEL software.

5.3 AD HOC DATABASE QUERIES

The basic operations that may be performed on a database table are retrieving rows and columns, inserting rows, deleting rows, and updating existing rows. In the SEL database, insertion, deletion, and update operations are all performed via DAMSEL, as described in the previous section. This is done to ensure that the semantic constraints imposed by the nature of the software engineering data, as discussed in Section 4.2, are enforced at all times. The operation of retrieving data, however, may be done in any context without risk of violating the integrity of the database. This section discusses how to perform database retrievals in an ad hoc manner. Additional examples of optimized SQL queries are presented in Appendix B. Although an introduction to the SQL SELECT statement is included, the coverage is not exhaustive. Refer to Reference 4 for a more in-depth presentation of the SQL language.

5.3.1 Connecting to the Database

Once a user with database access (Section 5.1) has logged onto the VAX, typing the following command at the system prompt connects him/her to the SEL database:

```
$ SQLPLUS
```

After supplying an ORACLE user ID and password at the prompts, the user is placed in an interpretive environment from which he/she may enter ad hoc SQL queries to retrieve database data. The command line prompt

SQL>

is displayed, signaling that the system is waiting for a SQL command. Upon entering a SQL command, terminated with a semicolon (;), and pressing the return key, SQL processes the command, displays the result, and returns to the SQL> prompt.

While in a SQL*Plus session, the following online HELP command is available:

SQL> HELP;

This displays a list of SQL commands, clauses, and related topics for which help is available.

To exit from a SQL*Plus session, the user types

SQL> EXIT

which will disconnect the user from ORACLE and return to the system prompt ($).

5.3.2 Basic Select Statement

The SQL statement for retrieving data from the database is the SELECT statement. In its simplest form, the SELECT statement has the following syntax:

SQL> SELECT * FROM <table-name>;

This statement displays on the terminal screen every row in the table indicated, as in the following example:

SQL> SELECT * FROM PROJECT;

PROJ_NAME	PROJ_NO	PROJ_TYPE	ACTIVE_STATUS
PROJ_101	101	SIMULATOR	ACT_DEV
PROJ_102	102	AGSS	ACT_DEV
PROJ_103	103	SIMULATOR	ACT_DEV
PROJ_104	104	SIMULATOR	ACT_DEV
PROJ_105	105	AGSS	ACT_DEV
PROJ_106	106	SIMULATOR	ACT_DEV
PROJ_71	71	SIMULATOR	INACTIVE
PROJ_110	110	AGSS	ACT_DEV
PROJ_108	108	SIMULATOR	ACT_DEV
PROJ_96	96	ORBIT	INACTIVE
PROJ_73	73	ATTITUDE	ACT_MAINT
PROJ_72	72	OTHER	ACT_DEV

The '*' in this form of the SELECT statement indicates that all columns of the table should be retrieved. To retrieve only specific columns, the '*' should be replaced by a list of the desired column names. The column names need not be specified in the order in which they are defined in the table definition, as illustrated in the following example:

```
SQL> SELECT PROJ_NO, PROJ_NAME FROM PROJECT;

PROJ_NO    PROJ_NAME
    108    PROJ_108
     96    PROJ_96
     73    PROJ_73
      .
      .
      .
```

5.3.3 Ordering the Retrieved Data

The SELECT statements seen thus far do not guarantee that the rows retrieved from the table will be displayed in any particular order. This may be ensured by specifying an ORDER BY clause on the SELECT statement, as in the following:

```
SQL> SELECT PROJ_NAME, PROJ_NO
  2  FROM   PROJECT
  3  ORDER  BY PROJ_NAME;

PROJ_NAME    PROJ_NO
PROJ_73           73
PROJ_101         101
PROJ_102         102
PROJ_110         110
      .
      .
      .
```

This causes the retrieved rows to be displayed in ascending order, sorted on the column specified in the ORDER BY clause. CHARACTER columns are sorted alphabetically, NUMBER columns are sorted numerically, and DATE columns are sorted chronologically. The default order in an ORDER BY clause is ascending. A display in descending order may be accomplished by specifying DESC after the name of the ORDER BY column. The ORDER BY clause also permits sorting on more than one field.

In the previous example, the SELECT statement was entered on more than one line. This illustrates that the SQL interpreter does not execute the command until a semicolon is entered. The typed command is stored in a buffer that is retained after the command is

executed. This buffer may be edited to change the query slightly without having to retype it completely. The current command in the buffer may be executed by typing

```
SQL> /
```

followed by a carriage return. The command buffer may be displayed by typing 'L', followed by a carriage return:

```
SQL> L
  1  SELECT  PROJ_NAME, PROJ_NO
  2  FROM    PROJECT
  3  ORDER   BY PROJ_NAME
```

Reference 4 provides details on editing the command buffer.

5.3.4 Limiting the Number of Rows Retrieved

The queries presented thus far have all displayed every row of the table specified. The WHERE clause allows constraints to be defined that limit the number of rows retrieved, as in the following example:

```
SQL> SELECT * FROM PROJECT WHERE PROJ TYPE = 'SIMULATOR';
```

PROJ_NAME	PROJ_NO	PROJ_TYPE	ACTIVE_STATUS
PROJ_101	101	SIMULATOR	ACT_DEV
PROJ_71	71	SIMULATOR	INACTIVE
PROJ_108	108	SIMULATOR	ACT_DEV
PROJ_103	103	SIMULATOR	ACT_DEV
PROJ_104	104	SIMULATOR	ACT_DEV
PROJ_106	106	SIMULATOR	ACT_DEV

This query selects only those records in which the PROJ TYPE column has a value of 'SIMULATOR'. It should be noted that, when specifying a character constant (or a date constant), it must be surrounded by single quotes. Date constants must be specified as follows: 'dd-mmm-yy', as in '05-JAN-88'. ORACLE character fields are case sensitive, and all the character fields in the SEL database that are commonly used in queries contain only uppercase characters.

Additional relational operators useful in specifying WHERE conditions include the following:

!=	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
IN	member of a list of items

The following example illustrates the use of the IN operator:

```
SQL> SELECT * FROM PROJECT
  2  WHERE PROJ_NO IN (101,103,105,107);
```

PROJ_NAME	PROJ_NO	PROJ_TYPE	ACTIVE_STATUS
PROJ_105	105	AGSS	ACT_DEV
PROJ_103	103	SIMULATOR	ACT_DEV
PROJ_101	101	SIMULATOR	ACT_DEV

Conditions in a WHERE clause may be combined by the logical connectives AND, OR, and NOT to build more complex conditions, as follows:

```
SQL> SELECT * FROM PROJECT
  2  WHERE PROJ_TYPE = 'SIMULATOR'
  3  AND   PROJ_NO > 104;
```

PROJ_NAME	PROJ_NO	PROJ_TYPE	ACTIVE_STATUS
PROJ_106	106	SIMULATOR	ACT_DEV
PROJ_108	108	SIMULATOR	ACT_DEV

When multiple conditions are specified, parentheses ( ) may be used to clarify or override precedence of operators.

5.3.5 Group Functions

A set of functions in SQL*Plus allows statistics to be calculated on the results of a query. Some of the most common of these are COUNT, AVG, MAX, MIN, SUM, STDDEV, and VARIANCE. The following example illustrates how these work:

```
SQL> SELECT  COUNT(PROJ_NO)
  2  FROM    PROJECT;

COUNT(PROJ_NO)
        90
```

This query returns a count of the number of rows in the PROJECT table that have a non-null value in the PROJ_NO column. Null values are entered into a particular column of a particular row to indicate that no data exist for that data item. The table definitions in Section 4.1 indicate which columns in the database will accept null values. Thus, in the case of the above query, since the PROJ_NO column does not accept null values, the query always returns a count of all rows in the table. Like COUNT, the statistical functions AVG, STDDEV, and VARIANCE operate only on non-null values. Another example is as follows:

```
SQL> SELECT  COUNT(RES_HR), SUM(RES_HR), AVG(RES_HR)
  2  FROM    PROJ_PROD
  3  WHERE   PROJ_NO = 151;
```

COUNT(RES_HR)	SUM(RES_HR)	AVG(RES_HR)
22	1.88	.085454545

5.3.6 Retrieving from More Than One Table—Joins

At this point, enough of the basic features of the SELECT statement have been presented to allow the user to find a particular piece of data in the database. Suppose, for example, the user wishes to know the names of the subsystem prefixes for project EXAMPLE. Consulting Section 4.3, the first step is to find the PROJ_NO value for that project:

```
SQL> SELECT PROJ_NO
  2  FROM   PROJECT
  3  WHERE PROJ_NAME = 'EXAMPLE';

PROJ_NO
   135
```

The user can use this result to retrieve the subsystem prefixes from PROJ_SUB:

```
SQL> SELECT SUB_PRE
  2  FROM   PROJ_SUB
  3  WHERE PROJ_NO = 135;

SUB_PRE
   PP
   SD
   TM
   PG
   CM
   UT
   AC
```

This works, but rather than doing this in two steps every time, the same result can be accomplished by a single query that **joins** the two tables:

```
SQL> SELECT SUB_PRE
  2  FROM   PROJECT, PROJ_SUB
  3  WHERE PROJ_NAME = 'EXAMPLE'
  4  AND    PROJECT.PROJ_NO = PROJ_SUB.PROJ_NO;

SUB_PRE
   PP
   SD
   TM
   PG
   CM
   UT
   AC
```

In this query, ORACLE created a virtual table containing all the columns in both the PROJECT and PROJ_SUB tables. If no constraints had been specified, the virtual table would have contained a row for each possible pairing of a row in PROJECT with a row in PROJ_SUB. However, the WHERE clause allowed it to create a virtual table in which the only row selected from the PROJECT table was that in which the PROJ_NAME was EXAMPLE; the only rows selected from the PROJ_SUB table were those in which the PROJ_NO column had the same value as the PROJ_NO column in the row selected from PROJECT (the PROJ_NO value for EXAMPLE). A join is not limited to two tables, and the columns displayed may come from any of the tables specified, as in the following example that displays the same subsystems as above, but includes the name of the project and the descriptive name of the subsystem:

```
SQL> SELECT  PROJ_NAME, SUB_PRE, NAME
  2  FROM    PROJECT, PROJ_SUB, SUBSYSTEM
  3  WHERE   PROJ_NAME = 'EXAMPLE'
  4  AND     PROJECT.PROJ_NO = PROJ_SUB.PROJ_NO
  5  AND     PROJ_SUB.SUBSY_ID = SUBSYSTEM.SUBSY_ID
  6  ORDER   BY SUB_PRE;
```

PROJ_NAME	SUB_PRE	NAME
EXAMPLE	AC	ATTITUDE AND ORBIT CONTROL
EXAMPLE	CM	COMMON BLOCKS
EXAMPLE	PG	PLOT GENERATOR

.
.
.

When the same column name occurs in more than one of the tables selected, that name must be qualified with the table name to refer to it within the query. Thus, PROJ_NO is qualified to differentiate between its occurrences in the PROJECT and PROJ_SUB tables, but PROJ_NAME need not be qualified, since it occurs only in the PROJECT table.

5.3.7 Retrieving from More Than One Table— Subqueries

Suppose the user wants to know the most recently estimated start and end dates for the design phase of project EXAMPLE. The user could join PROJECT and PROJ_EST_PHASE on the PROJ_NO field and get all of the estimated design phase start and end dates for that project. To limit the retrieval to only one pair of dates, however, a subquery is used. The most common use of a subquery is in specifying conditions on a WHERE clause, as follows:

```
SQL> SELECT  PROJ_NAME, PHASE_CO, START_DATE, END_DATE
  2  FROM    PROJECT, PROJ_EST_PHASE
  3  WHERE   PROJ_NAME = 'EXAMPLE'
  4  AND     PHASE_CO = 'DESGN'
  5  AND     PROJECT.PROJ_NO = PROJ_EST PHASE.PROJ_NO
```

```
4  6  AND     SUB_DATE =
   7          (SELECT MAX(SUB_DATE)
   8          FROM    PROJ_EST_PHASE
   9          WHERE   PROJ_EST_PHASE.PROJ_NO = PROJECT.PROJ_NO);
```

PROJ_NAME	PHASE_CO	START_DATE	END DATE
EXAMPLE	DESGN	06-JUN-87	02-JAN-88

This query joins the PROJECT and PROJ_EST_PHASE tables on the PROJ_NO field, and further limits the retrieval by specifying that only the PROJ_EST_PHASE row with the most recent SUB_DATE for the specified project be selected. Note that subqueries are enclosed in parentheses, and they must return a single value or a single column of values. The relational operator IN may be used to see if a value is in a column of values returned by a subquery. Also, subqueries may be nested, as in the following example that lists the names of all components under project EXAMPLE:

```
SQL> SELECT COM_NAME
  2  FROM   SUB_COM
  3  WHERE  SUBSY_ID IN
  4         (SELECT SUBSY_ID
  5         FROM    PROJ_SUB
  6         WHERE   PROJ_NO =
  7                 (SELECT PROJ_NO
  8                 FROM    PROJECT
  9                 WHERE   PROJ_NAME = 'EXAMPLE'));
```

```
COM_NAME
    PROID
    PROINI
    PROINT
    ACQINT
    DELP
    GETCAS
      .
      .
      .
```

5.3.8 Views—A Shortcut for Commonly Used Joins

Several views have been defined in the SEL database to allow users quick access to commonly used data items. A view is a virtual table that consists of columns from one or more tables selected by criteria specified in the definition of the view. For example, to be able to retrieve

all the component names for a given project, the V_PROJ_COM view was defined (refer to the table and view definitions in Section 4.1). Thus, the following:

```
SQL> SELECT  * FROM V_PROJ_COM
     WHERE   PROJ_NAME = <project name>;
```

is equivalent to

```
SQL> SELECT  PROJ_NAME, SUB_PRE, COM_NAME, COM_NO
     FROM    PROJECT, PROJ_SUB, SUB_COM
     WHERE   PROJ_NAME = <project name>
     AND     PROJECT.PROJ_NO = PROJ_SUB.PROJ_NO
     AND     PROJ_SUB.SUBSY_ID = SUB_COM.SUBSY_ID;
```

Similarly, the view V SUBSYSTEM INFO allows subsystem information to be selected using the following query:

```
SQL> SELECT  * FROM V_SUBSYSTEM_INFO
     WHERE   PROJ_NAME = <project name>;
```

This is equivalent to

```
SQL> SELECT  SUB_PRE, NAME, FUNCTION, SUB_DATE, PROJ_NAME
     FROM    PROJECT, PROJ_SUB, SUBSYSTEM
     WHERE   PROJ_NAME = <project name>
     AND     PROJECT.PROJ_NO = PROJ_SUB.PROJ_NO
     AND     PROJ_SUB.SUBSY_ID = SUBSYSTEM.SUBSY_ID;
```

Finally, the view V_PROJ_SUB_ACT is a shortcut to retrieve the activity hours charged to a particular subsystem. Thus,

```
SQL> SELECT  * FROM V_PROJ_SUB_ACT
     WHERE   PROJ_NAME = <project name>
     AND     SUB_PRE = <subsystem prefix>;
```

is equivalent to

```
SQL> SELECT  PROJ_NAME, SUB_PRE, ACTIVITY, ACT_HR
     FROM    PROJECT, EFF_PROJ, EFF_SUB, EFF_ACT
     WHERE   PROJ_NAME =<project name>
     AND     PROJECT.PROJ_NO = EFF_PROJ.PROJ_NO
     AND     EFF_PROJ.P_ID = EFF_SUB.P_ID
     AND     SUB_PRE = <subsystem prefix>
     AND     EFF_SUB.PS_ID = EFF_ACT.EFF_ID;
```

5.3.9 Spooling Output and Saving Queries

All the queries presented displayed their results on the terminal screen. To create a permanent copy of the query results, it is necessary to spool the query session, or at least part of it, to a file. This can be accomplished with the following command:

```
SQL> SPOOL <VMS file name>;
```

If no file extension is supplied as part of the file name, a file is created in the current default directory with the extension .LIS. After this command is entered, any queries executed and the associated results are written to this file, as well as displayed on the screen. Spooling can be turned off, with the following command:

SQL> SPOOL OFF;

It is also useful to save the contents of the current command buffer and reload it at some future time. The first step can be accomplished with the following commands:

SQL> SAVE <VMS file name>;

If no file extension is supplied as part of the file name, a file is created in the current default directory with the extension .SQL. This query can be reloaded into the command buffer by using the following command:

SQL> GET <VMS file name>;

This command searches the current default directory for the file name specified. If no extension is supplied in the file name, it searches for a file with extension .SQL. The loaded query may now be executed or listed with / or L as described in Section 5.3.3.

This section presents enough about ad hoc database queries to enable the user to access any particular item of software engineering data in which he or she is interested. It does not, however, cover all of the features in SQL*Plus that facilitate data retrieval. Some additional capabilities include displaying computed columns, simple pattern matching in WHERE clauses, conversion between data types, renaming column headings and defining display formats, parameterizing queries, computing statistics on groups of records, and printing them on break points when the value of a particular column changes. Readers who are interested in these and other advanced features should refer to Reference 4.

5.4 QUERY LIBRARY

A collection of commonly used, generalized queries is organized into a library on the STL VAX-11/780. The library includes a search facility with predefined commands to aid the users in locating appropriate queries to retrieve desired information. The queries are grouped into categories by the type of data they retrieve, as follows:

- Projects—General project data, statistics
- Effort—Personnel and services hours, activity hours
- Changes—Change and error data from CRFs
- Estimates—Estimated statistics and phase dates
- Growth—Growth history data
- Computers—Computer resource data

- Components—Component data from COFs
- Programmers—Programmer hours, activities
- Other—Miscellaneous queries not covered above

The search facility prompts for a category and provides a brief description of all queries available under that category. A help command is also available that provides instructions for using the library and lists the categories available.

Most of the queries prompt for parameters such as project name and date. The user should note the following two important constraints:

1. All character data must be typed in UPPER CASE
2. All dates must be entered in the format DD-MMM-YY (e.g., 01-JAN-89)

Once a user with database access (Section 5.1) has logged onto the VAX, the following command is typed to connect to SQL*Plus:

```
$ SQLPLUS
```

After supplying an ORACLE user ID and password at the prompts, the user is placed in an interpretive environment from which he or she may use the query library. The command line prompt

```
SQL>
```

is displayed, signaling that the system is waiting for a SQL command. Online query library help is available by typing

```
SQL> START QLIB:QHELP
```

NOTE: The symbol "@" can be used in place of the word "START" (i.e., @QLIB:QHELP)

The available help information on the query library will be displayed. To view a list of available queries and their associated description, type the following:

```
SQL> START QLIB:SEARCH
```

The user will be prompted for the name of one of the above categories.

If the user is unsure of the category names, he or she should type a question mark (?) and all categories will be listed. Once the desired query has been located, the query can be executed by typing

```
SQL> START QLIB:<query name>
```

All requested parameters should then be entered (note the previously mentioned constraints). If the user wants to save the result, the following steps should be executed:

```
SQL> SPOOL <output file>
SQL> START QLIB:<query name>
SQL> SPOOL OFF
```

The output will be located in user's directory and appear as /output file/.LIS. Once the user has completed use of the library, he/she can enter ad hoc queries (Section 5.3) or exit from SQL*Plus by typing

```
SQL> EXIT
```

The system prompt will be displayed.

APPENDIX A—ENCODED FIELDS AND ALLOWABLE VALUES

This appendix lists all the codes used throughout the SEL database and their corresponding values. Items are listed alphabetically according to the field in which the code is stored. Exceptions to this are CL_ACTIVITY, DATA_AVAIL, and QA_STATUS. The CL_ACTIVITY codes are the Cleanroom PRF values that are stored in the ACTIVITY field of the EFF_ACT table. DATA_AVAIL and QA_STATUS codes are stored only in the VALIDATION table, but are included in the VAL_DATA_AVAIL and VAL_QA_STATUS views, respectively.

Fields Where Used	Code	Value (Description)
ACTIVE_STATUS	ACT_DEV	Data collection is active; project is in development
ACTIVE_STATUS	ACT_MAINT	Data collection is active; project is in maintenance
ACTIVE_STATUS	DISCONT	Data collection discontinued; data for the project are incomplete; no plan to validate data
ACTIVE_STATUS	INACTIVE	The project has been completed and no more data are being collected
ACTIVITY	ACCTEST	Acceptance test
ACTIVITY	CREDES	Create design
ACTIVITY	DEBUG	Debugging
ACTIVITY	INTTEST	Integration test
ACTIVITY	OTHER	Other
ACTIVITY	PREDES	Predesign
ACTIVITY	RDREVCOD	Read/review code
ACTIVITY	RDREVDES	Read/review design
ACTIVITY	SUPPORT	Support
ACTIVITY	TSTCODUN	Test code units
ACTIVITY	WRCODE	Write code
ADA_FEATURE	DATATYPE	Data typing

Fields Where Used	Code	Value (Description)
ADA_FEATURE	EXCEPT	Exceptions
ADA_FEATURE	GEN	Generics
ADA_FEATURE	OTHER	Other
ADA_FEATURE	PACK	Program structure and packaging
ADA_FEATURE	SUBPROG	Subprograms
ADA_FEATURE	SYSDEPF	System dependent features
ADA_FEATURE	TASK	Tasking
CH_CAUSE	CODE	Code
CH_CAUSE	DESIGN	Software Design
CH_CAUSE	OTHER	Other
CH_CAUSE	PRECH	Previous Change
CH_CAUSE	REQMTSPEC	Requirements/functional specifications
CH_CLASS	COMPUTE	Computational
CH_CLASS	DATAVAL	Data (value or structure)
CH_CLASS	INIT	Initialization
CH_CLASS	INTERE	Interface (external)
CH_CLASS	INTERI	Interface (internal)
CH_CLASS	LOGIC	Logic/control structure
CH_CLASS	OTHER	Other
CH_OBJECT	CODE	Code
CH_OBJECT	DESIGNDOC	Design document
CH_OBJECT	OTHER	Other
CH_OBJECT	REQMTDOC	Requirements/specifications document
CH_OBJECT	SYSDESC	System description
CH_OBJECT	USERGUIDE	User's guide
CH_TYPE	ADENC	Adaptation to environment change
CH_TYPE	ERRCO	Error correction

Fields Where Used	Code	Value (Description)
CH_TYPE	IMPCM	Improvement of clarity, maintainability, or documentation
CH_TYPE	IMPRE	Implementation of requirements change
CH_TYPE	IMPUS	Improvement of user services
CH_TYPE	IN/DE	Insertion/deletion of debug code
CH_TYPE	OPTSA	Optimization of time/space/accuracy
CH_TYPE	OTHCH	Other change type
CH_TYPE	PLANE	Planned enhancement
CL_ACTIVITY	CLACCTEST	Cleanroom acceptance test
CL_ACTIVITY	CLCREDES	Cleanroom system, subsystems, or components design
CL_ACTIVITY	CLINDTEST	Cleanroom system components testing by independent tester
CL_ACTIVITY	CLOTHER	Cleanroom other hours, i.e., management, meetings, documentation, etc.
CL_ACTIVITY	CLPREDES	Cleanroom predesign, such as requirements analysis
CL_ACTIVITY	CLPRETEST	Cleanroom pretest
CL_ACTIVITY	CLRDREVCOD	Cleanroom code read and code verification
CL_ACTIVITY	CLRESPSFR	Cleanroom response to tester reported problems and solution implementation
CL_ACTIVITY	CLVEREVDES	Cleanroom design verification and review, including meetings, reviews, or walkthroughs
CL_ACTIVITY	CLWRCODE	Cleanroom system components coding
CL_ACTIVITY	SUPPORT	Cleanroom support
COM_TYPE	ADAGENB	Ada generic body
COM_TYPE	ADAGENS	Ada generic specification
COM_TYPE	ADAPACKB	Ada package body

Fields Where Used	Code	Value (Description)
COM_TYPE	ADAPACKS	Ada package specification
COM_TYPE	ADASUBB	Ada subprogram body
COM_TYPE	ADASUBS	Ada subprogram specification
COM_TYPE	ADATASKB	Ada task body
COM_TYPE	ADATASKS	Ada task specification
COM_TYPE	ADAUNSPEC	Ada source code (type unspecified)
COM_TYPE	ALC	Assembly language component
COM_TYPE	BLOCKDA	BLOCK DATA component
COM_TYPE	DISPALY	Dispaly identification
COM_TYPE	FORTRAN	FORTRAN source code
COM_TYPE	INCL	Include file
COM_TYPE	JCL	Job control language
COM_TYPE	MENDEF	Menu definition or help file
COM_TYPE	NAMELT	NAMELIST or parameter list
COM_TYPE	OTHER	Other type of component
COM_TYPE	PASCAL	Pascal source code
COM_TYPE	REFDATA	Reference data file
DATA_AVAIL	COF	Component origination information available
DATA_AVAIL	COM_NAME	Component names available
DATA_AVAIL	CPU	Project computer resources available
DATA_AVAIL	CRF	Component change information available
DATA_AVAIL	EFF_PROJ	Manpower effort data at the project level available
DATA_AVAIL	EFF_SERV	Services effort data (Tech. Pubs., Secretary, etc.) available
DATA_AVAIL	EFF_SPEC	Manpower effort data for special activities (rework, reuse, etc.) available

Fields Where Used	Code	Value (Description)
DATA_AVAIL	EFF_SUB	Manpower effort data at the subsystem level available
DATA_AVAIL	EST_SCH	Estimated project phase schedules available
DATA_AVAIL	EST_STAT	Estimated project statistics (LOC, effort data, component data) available
DATA_AVAIL	FIN_CPU	Closed project—Final computer resources available
DATA_AVAIL	FIN_SCH	Closed project—Final phase dates available
DATA_AVAIL	FIN_STAT	Closed project—Final statistics (LOC, effort, component data) available
DATA_AVAIL	GRH	Project growth data available
DATA_AVAIL	SAP	Closed project—Detailed component analysis available
DATA–AVAIL	SEF	Close project—Sujective evaluation data available
DATA–AVAIL	SIF	Subsystem information available
EFF_COM_CH	1HR	1 hour or less
EFF_COM_CH	1DAY	1 hour to 1 day
EFF_COM_CH	3DAY	1 day to 3 days
EFF_COM_CH	NDAY	More than 3 days
EFF_COM_CH	NOTDET	Not determined
EFF_ISO_CH	1HR	1 hour or less
EFF_ISO_CH	1DAY	1 hour to 1 day
EFF_ISO_CH	3DAY	1 day to 3 days
EFF_ISO_CH	NDAY	More than 3 days
EFF_ISO_CH	NOTDET	Not determined
ERR_ACAUSE	FEATUREC	Confused features
ERR_ACAUSE	FEATUREM	Misunderstood features

Fields Where Used	Code	Value (Description)
ERR_ACAUSE	INCOF	Features applied incorrectly
ERR_ACAUSE	INTERACT	Misunderstood interaction of features
ERR_ARES	MEMORY	Own memory
ERR_ARES	NOTE	Class notes
ERR_ARES	NTEAM	Someone not on project team
ERR_ARES	OTHER	Other
ERR_ARES	REFMAN	Ada reference manual
ERR_ARES	TEAM	Own project team member
ERR_CLASS	COMPUTE	Computational
ERR_CLASS	DATAVAL	Data value or structure
ERR_CLASS	INIT	Initialization
ERR_CLASS	INTERE	Interface (external)
ERR_CLASS	INTERI	Interface (internal)
ERR_CLASS	LOGIC	Logic/control structure
ERR_CLASS	NOTDET	Not determined
ERR_SOURCE	CODE	Code
ERR_SOURCE	DESIGN	Design
ERR_SOURCE	FUNSPEC	Functional specifications
ERR_SOURCE	NOTDET	Not determined
ERR_SOURCE	PRECH	Previous change
ERR_SOURCE	REQMT	Requirements
ERR_TOOLS	CMS	Code Management System
ERR_TOOLS	COMPI	Compiler
ERR_TOOLS	DECTM	DEC Test Manager
ERR_TOOLS	LSE	Language sensitive editor
ERR_TOOLS	OTHER	Other
ERR_TOOLS	PCA	Performance and coverage analyzer

Fields Where Used	Code	Value (Description)
ERR_TOOLS	SCA	Source code analyzer
ERR_TOOLS	SYMDEB	Symbolic debugger
FINAL_ORIGIN_CAT	EXTMO	Extensively modified
FINAL_ORIGIN_CAT	NEW	Completely new
FINAL_ORIGIN_CAT	OLDUC	Old (unchanged)
FINAL_ORIGIN_CAT	SLMOD	Slightly modified
FUNCTION	CPEXEC	Control processing/executive
FUNCTION	DPDC	Data processing/data conversion
FUNCTION	GRAPH	Graphics and special device support
FUNCTION	MATHCOMP	Mathematical/computational
FUNCTION	REALTIME	Real-time control
FUNCTION	SYSSERV	System services
FUNCTION	USERINT	User interface
MAINT_ACT	ACCBENTEST	Hours spend on acceptance/benchmark testing
MAINT_ACT	IMPLEMENT	Hours spend on changing a system, code and the associated documentation included
MAINT_ACT	ISOLATION	Hours spend on understanding the failure or request for enhancement of adaptation
MAINT_ACT	OTHER	Hours spend on other maintenance activities
MAINT_ACT	REDESIGN	Hours spent on redesigning a system
MAINT_ACT	UNSYSTEST	Hours spend on unit/system testing
MAINT_CH_TYPE	ADAPTATION	Adaptation (response to change of operational environment)
MAINT_CH_TYPE	CORRECTION	Correction (system did not satisfy its requirements)

Fields Where Used	Code	Value (Description)
MAINT_CH_TYPE	ENHANCEMNT	Enhancement (response to change of requirements)
MAINT_CLASS	ADAPTATION	Hours spend on maintenance with modifying a system to adapt to a change
MAINT_CLASS	CORRECTION	Hours spend on maintenance with a system failure
MAINT_CLASS	ENHANCEMNT	Hours spent on maintenance with a system failure
MAINT_CLASS	OTHER	Hours spent on other maintenance activities
MAINT_COM_CH	1HR	1 hour or less
MAINT_COM_CH	1DAY	1 hour to 1 day
MAINT_COM_CH	1WEEK	1 day to 1 week
MAINT_COM_CH	1MONTH	1 week to 1 month
MAINT_COM_CH	1MONTHMORE	More than 1 month
MAINT_ISO_CH	1HR	1 hour or less
MAINT_ISO_CH	1DAY	1 hour to 1 day
MAINT_ISO_CH	1WEEK	1 day to 1 week
MAINT_ISO_CH	1MONTH	1 week to 1 month
MAINT_ISO_CH	1MONTHMORE	More than 1 month
MEASURE_CODE	ACCTSTONE	Number of acceptance tests executed at least one time
MEASURE_CODE	ACCTSTPASS	Number of acceptance tests passed
MEASURE_CODE	ACCTSTRUN	Number of acceptance test runs, including reruns
MEASURE_CODE	DISCRES	Number of discrepancies resolved
MEASURE_CODE	MODCODE	Number of modules completed
MEASURE_CODE	MODDESIGN	Number of modules designed

Fields Where Used	Code	Value (Description)
MEASURE_CODE	QUESTANS	Number of questions answered by analysts
MEASURE_CODE	SPECMODIMP	Number of specification modifications implemented
MEASURE_CODE	SYSTSTONE	Number of system tests executed at least one time
MEASURE_CODE	SYSTSTPASS	Number of system tests passed
MEASURE_CODE	SYSTSTRUN	Number of system test runs, including reruns
MEAS_TYPE	PM01	Problem difficulty
MEAS_TYPE	PM02	Tightness of schedule constraints
MEAS_TYPE	PM03	Requirements stability
MEAS_TYPE	PM04	Quality of specification documents
MEAS_TYPE	PM05	Requirements for documentation
MEAS_TYPE	PM06	Rigor of formal reviews
MEAS_TYPE	ST07	Ability of development team
MEAS_TYPE	ST08	Development team experience with application
MEAS_TYPE	ST09	Development team experience with environment
MEAS_TYPE	ST10	Stability of development team composition
MEAS_TYPE	TM11	Project management performance
MEAS_TYPE	TM12	Project management experience with application
MEAS_TYPE	TM13	Stability of project management team
MEAS_TYPE	TM14	Project planning discipline
MEAS_TYPE	TM15	Degree project plans followed
MEAS_TYPE	PC16	Modern programming practices
MEAS_TYPE	PC17	Disciplined change/question tracking

Fields Where Used	Code	Value (Description)
MEAS_TYPE	PC18	Use of disciplined requirements analysis methodology
MEAS_TYPE	PC19	Use of disciplined design methodology
MEAS_TYPE	PC20	Use of disciplined testing methodology
MEAS_TYPE	PC21	Use of tools
MEAS_TYPE	PC22	Use of test plans
MEAS_TYPE	PC23	Use of quality assurance procedures
MEAS_TYPE	PC24	Use of configuration management procedures
MEAS_TYPE	EN25	Degree of access to development system
MEAS_TYPE	EN26	Programmers per terminal
MEAS_TYPE	EN27	Development machine resource constraints
MEAS_TYPE	EN28	System response time
MEAS_TYPE	EN29	System hardware and support software stability
MEAS_TYPE	EN30	Software tool effectiveness
MEAS_TYPE	PT31	Delivered software supports requirements
MEAS_TYPE	PT32	Quality of delivered software
MEAS_TYPE	PT33	Quality of design present in delivered software
MEAS_TYPE	PT34	Quality/completeness of software documentation
MEAS_TYPE	PT35	Timely software delivery
MEAS_TYPE	PT36	Smoothness of acceptance testing
NOTE_TYPE	CLOSEOUT	Project closeout status
NOTE_TYPE	COMPACCTS	Computer accounts to monitor

Fields Where Used	Code	Value (Description)
NOTE_TYPE	COMPSYS	Development and operational computer system
NOTE_TYPE	CONTACTS	Project contacts
NOTE_TYPE	CONTRLLIB	Names of controlled libraries
NOTE_TYPE	DATAAVAIL	Type of data available
NOTE_TYPE	FORMSCOL	SEL forms collected
NOTE_TYPE	GENMESS	General messages
NOTE_TYPE	GHTOOL	Growth history tool used
NOTE_TYPE	LANGUAGES	Languages used
NOTE_TYPE	PROJNAME	Project full name
NOTE_TYPE	TASKNO	Task numbers and corresponding years
ORI_TYPE	EXTMO	Extensively modified
ORI_TYPE	NEW	Completely new
ORI_TYPE	OLDUC	Old (unchanged)
ORI_TYPE	SLMOD	Slightly modified
PHASE_CO	ACCTE	Acceptance test
PHASE_CO	CLEAN	Cleanup
PHASE_CO	CODET	Code and test (implementation)
PHASE_CO	DESGN	Design
PHASE_CO	MAINT	Maintenance
PHASE_CO	REQNT	Requirement definition
PHASE_CO	SYSTE	System test
PROJ_TYPE	AGSS	Attitude ground support system
PROJ_TYPE	ATTITUDE	Attitude oriented
PROJ_TYPE	DATABASE	Database
PROJ_TYPE	GRAPH/UI	Graphics/user interface
PROJ_TYPE	MP&A	Mission planning and analysis

Fields Where Used	Code	Value (Description)
PROJ_TYPE	ORBIT	Orbit oriented
PROJ_TYPE	OTHER	Other
PROJ_TYPE	REALTIME	Real time processing
PROJ_TYPE	SIMULATOR	Simulator
PROJ_TYPE	TOOL	Software tool
PURPOSE	ADADA	Ada data abstraction
PURPOSE	ADAPR	Ada process abstraction
PURPOSE	ALCOMP	Algorithmic/computational
PURPOSE	CNTRMOD	Control module
PURPOSE	DATRA	Data transfer
PURPOSE	INTOP	Interface to operating system
PURPOSE	IOPRO	I/O processing
PURPOSE	LODEC	Logic/decision
QA_STATUS	HCCORRECT	Hand-checked: correct
QA_STATUS	HCERROR	Hand-checked: errors found
SECOND_L	CAT	Configuration Analysis Tool
SECOND_L	CMTOOL	Configuration management tool (e.g. CMS, MMS)
SECOND_L	COMPI	Compiler
SECOND_L	EDIT	Editor
SECOND_L	GRADIS	Graphics display builder
SECOND_L	INTERF	Interface checker (e.g., RXVP80, ANALYZ)
SECOND_L	ISPF	ISPF
SECOND_L	LINK	Linker
SECOND_L	LSE	Language sensitive editor
SECOND_L	OTHER	Other tools
SECOND_L	PANVAL	PANVALET

Fields Where Used	Code	Value (Description)
SECOND_L	PDLPR	PDL processor
SECOND_L	REPLP	Requirement language processor
SECOND_L	SAP	Source Code Analyzer program
SECOND_L	SDE	Software development environment
SECOND_L	STRANT	Structured analysis tool
SECOND_L	SYMDEB	Symbolic debugger
SECOND_L	TESTCO	Test coverage tool
SP_ACTIVITY	CLMETHOD	Methodology understanding or discussion
SP_ACTIVITY	DOCUMENT	Document
SP_ACTIVITY	ENHANCE	Enhance/refine/optimize
SP_ACTIVITY	REUSE	Reuse
SP_ACTIVITY	REWORK	Rework
STATUS	CLOSED	Information has been verified and validated—Project is closed
STATUS	HCCORRECT	Hand-checked: correct
STATUS	HCERROR	Hand-checked: errors found
STATUS	UNCHK	Unchecked
STATUS	VERAP	Verified by application
STATUS_CODE	ACCTST	Acceptance testing status
STATUS_CODE	CODE	Code status
STATUS_CODE	DESIGN	Design status
STATUS_CODE	DISCREP	Discrepancy status
STATUS_CODE	QUESTIONS	Questions to analysts status
STATUS_CODE	SPECMOD	Specification modification status
STATUS_CODE	SYSTST	System testing status
TARGET_CODE	QUESTSUB	Number of questions submitted to analysts

Fields Where Used	Code	Value (Description)
TARGET_CODE	SPECMODREC	Number of specification modifications received
TARGET_CODE	TOTACCTST	Total number of separate acceptance tests planned
TARGET_CODE	TOTCODE	Estimated total number of modules to be coded
TARGET_CODE	TOTDESIGN	Estimated total number of modules to be designed
TARGET_CODE	TOTDISCREP	Total number of discrepancies reported
TARGET_CODE	TOTSYSTST	Total number of separate system tests planned

APPENDIX B—SAMPLE OPTIMIZED DATABASE QUERIES

This appendix contains additional examples of SQL queries to augment those presented in Section 5.3. These are optimized queries that are written specifically for an ORACLE RDBMS environment. In each example, the desired retrieval is first expressed in an English statement. This is followed by SQL statements to retrieve the desired data. The user should remember that there is often more than one way to formulate a particular query; only one method is presented here for each example.

1. Retrieve the names of all Attitude Ground Support Systems (AGSSs) with more than 100,000 total lines of code.

```
SQL> SELECT  PROJ_NAME
     FROM    PROJ_STAT,  PROJECT
     WHERE   T_LINE > 100000
     AND     PROJ_TYPE = 'AGSS'
     AND     PROJECT.PROJ_NO = PROJ_STAT.PROJ_NO;
```

2. Retrieve the names of all persons who have submitted PRFs for project 'XYZ'.

```
SQL> SELECT  DISTINCT FULL_NAME
     FROM    EFF_FORM, EFF_PROJ, PERSONNEL, PROJECT
     WHERE   FORM_TYPE = 'PRF'
     AND     EFF_PROJ.P_ID = EFF_FORM.P_ID
     AND     EFF_PROJ.PROG_ID = PERSONNEL.PROG_ID
     AND     EFF_PROJ.PROJ_NO = PROJECT.PROJ_NO
     AND     PROJ_NAME = 'XYZ';
```

3. For project 'XYZ', list alphabetically all component names (with subsystem prefixes) that do not have COF data.

```
SQL> SELECT  SUB_PRE, COM_NAME
     FROM    V_PROJ_COM
     WHERE   PROJ_NAME = 'XYZ'
     AND     COM_NO NOT IN
             (SELECT COM_NO FROM COM_SOURCE)
     ORDER   BY SUB_PRE, COM_NAME;
```

4. Retrieve the number of error correction changes for project 'XYZ' that took more than 3 days to implement.

```
SQL> SELECT  COUNT (CHANGE_NO)
     FROM    CHANGE
     WHERE   CHANGE_NO IN
             (SELECT DISTINCT CHANGE _NO
             FROM    CHANGE_COM, V_PROJ_COM
             WHERE   CHANGE_COM.COM_NO =
                     V_PROJ_COM.COM_NO
             AND     PROJ_NAME = 'XYZ')
     AND     EFF_COM_CH = 'NDAY'
     AND     CH_TYPE = 'ERRCO';
```

5. Retrieve the total design hours for project 'XYZ'. This query may be interpreted two ways.

   a. Retrieve all hours charged to design activities.

```
SQL> SELECT  SUM (ACT_HR)
     FROM    EFF_ACT
     WHERE   EFF_ID IN
             (SELECT P_ID
             FROM    EFF_PROJ, PROJECT
             WHERE   EFF_PROJ.PROJ_NO =
                     PROJECT.PROJ_NO
             AND     PROJ_NAME = 'XYZ'
             UNION
             SELECT  PS_ID
             FROM    EFF_SUB, EFF_PROJ, PROJECT
             WHERE   EFF_PROJ.P_ID = EFF_SUB.P_ID
             AND     EFF_PROJ.PROJ_NO = PROJECT.PROJ_NO
             AND     PROJ_NAME = 'XYZ')
             AND     ACTIVITY IN ('CREDES', 'RDREVDES');
```

   b. Retrieve all manpower hours charged during the design phase.

   First, find the design phase start and end dates.

```
SQL> SELECT  START_DATE, END_DATE
             PROJ_EST_PHASE, PROJECT
     WHERE   SUB_DATE =
             (SELECT MAX (SUB_DATE)
             FROM    PROJ_EST_PHASE
             WHERE   PROJ_NO = PROJECT.PROJ_NO)
```

```
AND     PHASE_CO = 'DESIGN'
AND     PROJ_EST_PHASE.PROJ_NO =
        PROJECT.PROJ_NO
AND     PROJ_NAME = 'XYZ'
```

Second, find all activity hours between these dates

```
SQL> SELECT  SUM (ACT_HR)
     FROM    EFF_ACT
     WHERE   EFF_ID IN
             (SELECT P_ID
             FROM    EFF_PROJ, PROJECT
             WHERE   SUB_DATE BETWEEN <start date>
             AND     <end date>
             AND     EFF_PROJ.PROJ_NO = PROJECT.PROJ_NO
             AND     PROJ_NAME = 'XYZ'
             UNION
             SELECT  PS_ID
             FROM    EFF_SUB, EFF_PROJ, PROJECT
             WHERE   SUB_DATE BETWEEN <start date>
             AND     <end date>
             AND     EFF_PROJ.P_ID = EFF_SUB.P_ID
             AND     EFF_PROJ.PROJ_NO = PROJECT.PROJ_NO
             AND     PROJ_NAME = 'XYZ'
             AND     ACTIVITY ! = 'SUPPORT');
```

APPENDIX C—SEL DATA COLLECTION FORMS

This appendix contains all the SEL data collection forms. Most forms are completed by programmers and managers of SEL-monitored projects. The PCSF, PMF, PSF, and SPF are completed by SEL personnel.

CHANGE REPORT FORM

Name: D21 Approved by: _____

Project: D1 Date: D2

Section A – Identification

Describe the change: (What, why, how) _____

Effect: What components are changed?

Prefix	Name	Version
D61	D62	

(Attach list if more space is needed)

Effort: What additional components were examined in determining what change was needed?

Location of developer's source files _____

		month	day	year
Need for change determined on:	D63			
Change completed (incorporated into system):	D64			

Check here if change involves Ada components (If so, complete questions on reverse side) ☐ D76

		1 hr/less	1 hr/1 day	1/3 days	>3 days
Effort in person time to isolate the change (or error):	D65				
Effort in person time to implement the change (or correction):	D66				

Section B – All Changes

Type of Change (Check one)

- ☐ Error correction
- ☐ Planned enhancement
- ☐ Implementation of requirements change
- ☐ Improvement of clarity, maintainability, or documentation
- ☐ Improvement of user services
- ☐ Insertion/deletion of debug code
- ☐ Optimization of time/space/ accuracy
- ☐ Adaptation to environment change
- ☐ Other (Describe below)

D67

Effects of Change

Y	N		
☐	☐	Was the change or correction to one and only one component? (Must match Effect in Section A)	D68
☐	☐	Did you look at any other component? (Must match Effort in Section A)	D69
☐	☐	Did you have to be aware of parameters passed explicitly or implicitly (e.g., COMMON blocks) to or from the changed components?	D70

Section C – For Error Corrections Only

Source of Error (Check one)

- ☐ Requirements
- ☐ Functional specifications
- ☐ Design
- ☐ Code
- ☐ Previous change

D71

Class of Error (Check most applicable)*

- ☐ Initialization D72
- ☐ Logic/control structure (e.g., flow of control incorrect)
- ☐ Interface (internal) (module-to-module communication)
- ☐ Interface (external) (module to external communication)
- ☐ Data (value or structure) (e.g., wrong variable used)
- ☐ Computational (e.g., error in math expression)

*If two are equally applicable, check the one higher on the list.

Characteristics (Check Y or N for all)

Y	N		
☐	☐	Omission error (e.g., something was left out)	D73
☐	☐	Commission error (e.g., something incorrect was included)	D74
☐	☐	Error was created by transcription (clerical)	D75

For Librarian's Use Only

Number: D82

Date: _____

Entered by: _____

Checked by: _____

10004437-g019

NOVEMBER 1991

Figure C-1. Change Report Form (CRF) (1 of 2)

CHANGE REPORT FORM
Ada Project Additional Information

1. Check which Ada feature(s) was involved in this change (Check all that apply)

D77

- ☐ Data typing
- ☐ Subprograms
- ☐ Exceptions
- ☐ Generics
- ☐ Program structure and packaging
- ☐ Tasking
- ☐ System-dependent features
- ☐ Other, please specify_____ (e.g., I/O, Ada statements)

2. For an error involving Ada components:

   a. Does the compiler documentation or the language reference manual explain the feature clearly? D78 _____ (Y/N)

   b. Which of the following is most true? (Check one)

D79

- ☐ Understood features separately but not interaction
- ☐ Understood features, but did not apply correctly
- ☐ Did not understand features fully
- ☐ Confused feature with feature in another language

   c. Which of the following resources provided the information needed to correct the error? (Check all that apply)

D80

- ☐ Class notes
- ☐ Ada reference manual
- ☐ Own project team member
- ☐ Own memory
- ☐ Someone not on team
- ☐ Other

   d. Which tools, if any, aided in the detection or correction of this error? (Check all that apply)

D81

- ☐ Compiler
- ☐ Symbolic debugger
- ☐ Language-sensitive editor
- ☐ CMS
- ☐ Source Code Analyzer
- ☐ P&CA (Performance and Coverage Analyzer)
- ☐ DEC test manager
- ☐ Other, specify_____

3. Provide any other information about the interaction of Ada and this change that you feel might aid in evaluating the change and using Ada

10004437-g020

NOVEMBER 1991

Figure C-1. Change Report Form (CRF) (2 of 2)

COMPONENT ORIGINATION FORM

Identification

Name: _____ D21

Project: _____ D1 Date: _____ D2

Subsystem Prefix: _____ D50

Component Name: _____ D53

Configuration Management Information

Date entered into controlled library (supplied by configuration manager): _____ D54

Library or directory containing developer's source file: _____

Member name: _____

Relative Difficulty of Developing Component D55

Please indicate your judgment by circling one of the numbers below.

Easy		Medium		Hard
1	2	3	4	5

Origin D56

If the component was modified or derived from a different project, please indicate the approximate amount of change and from where it was acquired; if it was coded new (from detailed design) indicate NEW.

_____ NEW
_____ Extensively modified (more than 25% of statements changed)
_____ Slightly modified
_____ Old (unchanged)

For Librarian's Use Only

Number: _____
Date: _____
Entered by: _____
Checked by: _____

If not new, what project or library is it from? _____
Component or member name: _____

Type of Component (Check one only) D57

_____ INCLUDE file (e.g., COMMON)
_____ Control language (e.g., JCL, DCL, CLIST)
_____ ALC (assembler code)
_____ FORTRAN source
_____ Pascal source
_____ C source
_____ NAMELIST or parameter list
_____ Display identification (e.g., GESS, FDAF)
_____ Menu definition or help
_____ Reference data files
_____ BLOCK DATA file
_____ Ada subprogram specification
_____ Ada subprogram body
_____ Ada package specification
_____ Ada package body
_____ Ada task body
_____ Ada generic instantiation
_____ Ada generic specification
_____ Ada generic body
_____ Other

Purpose of Executable Component D58

For executable code, please identify the major purpose or purposes of this component. (Check all that apply).

_____ I/O processing
_____ Algorithmic/computational
_____ Data transfer
_____ Logic/decision
_____ Control module
_____ Interface to operating system
_____ Process abstraction
_____ Data abstraction

10004437 g021

NOVEMBER 1991

Figure C-2. Component Origination Form (COF)

DEVELOPMENT STATUS FORM

Name: D21

Project: D1 Date: D22

Please complete the section(s) that is appropriate for the current status of the project.

Design Status	
Planned total number of components to be designed (New, modified, and reused)	D180
Number of components designed (Prolog and PDL have been completed)	D181

Code Status	
Planned total number of components to be coded (New, modified, and reused)	D182
Number of components completed (Added to controlled library)	D183

Testing Status	System Test	Acceptance Test
Total number of separate tests planned	D184	D188
Number of tests executed at least one time	D185	D189
Number of tests passed	D186	D190

Discrepancy Tracking Status (from beginning of system testing)	
Total number of discrepancies reported	D192
Total number of discrepancies resolved	D193

Specification Modification Status (from beginning of requirements analysis)	
Total number of specification modifications received	D194
Total number of specification modifications completed (implemented)	D195

Requirements Questions Status (from beginning of requirements analysis)	
Total number of questions submitted to analysts	D196
Total number of questions answered by analysts	D197

Check here if there are no changes ☐

For Librarian's Use Only
Number: D198
Date:
Entered by:
Checked by:

10004437-g022

NOVEMBER 1991

Figure C-3. Development Status Form (DSF)

MAINTENANCE CHANGE REPORT FORM

For Librarian's Use Only
Number: D178
Date:
Entered by:
Checked by:

Name: D21 OSMR Number: D162

Project: D1 Date: D2

SECTION A: Change Request Information

Functional Description of Change: _____

What was the type of modification? D163

- ___ Correction
- ___ Enhancement
- ___ Adaptation

What caused the change? D164

- ___ Requirements/specifications
- ___ Software design
- ___ Code
- ___ Previous change
- ___ Other

SECTION B: Change Implementation Information

Components Added/Changed/Deleted: _____

	< 1hr	1 hr to 1 day	1 day to 1 week	1 week to 1 month	> 1 month	
Estimate effort spent isolating/determining the change:						D165
Estimate effort to design, implement, and test the change:						D166

Check all changed objects: D167

- ___ Requirements/Specifications Document
- ___ Design Document
- ___ Code
- ___ System Description
- ___ User's Guide
- ___ Other

If code changed, characterize the change (check most applicable): D168

- ___ Initialization
- ___ Logic/control structure (e.g., changed flow of control)
- ___ Interface (internal) (module to module communication)
- ___ Interface (external) (module-to-external communication)
- ___ Data (value or structure) (e.g., variable or value changed)
- ___ Computational (e.g., change of math expression)
- ___ Other (none of the above apply)

Estimate the number of lines of code (including comments): D169 added D170 changed D171 deleted

Enter the number of components: D172 added D173 changed D174 deleted

Enter the number of the added components that are: D175 totally new D176 totally reused D177 reused with modifications

10004437-g023

NOVEMBER 1991

Figure C-4. Maintenance Change Report Form (MCRF)

Personnel Resources Form

Name: D21

Project: D1 Date (Friday): D22

SECTION A: Total Hours Spent on Project for the Week: _____

SECTION B: Hours By Activity (Total of hours in Section B should equal total hours in Section A)

Activity	Activity Definitions	Hours
Predesign	Understanding the concepts of the system. Any work prior to the actual design (such as requirements analysis).	D23
Create Design	Development of the system, subsystem, or components design. Includes development of PDL, design diagrams, etc.	D24
Read/Review Design	Hours spent reading or reviewing design. Includes design meetings, formal and informal reviews, or walkthroughs.	D25
Write Code	Actually coding system components. Includes both desk and terminal code development.	D26
Read/Review Code	Code reading for any purpose other than isolation of errors.	D27
Test Code Units	Testing individual components of the system. Includes writing test drivers.	D28
Debugging	Hours spent finding a known error in the system and developing a solution. Includes generation and execution of tests associated with finding the error.	D29
Integration Test	Writing and executing tests that integrate system components, including system tests.	D30
Acceptance Test	Running/supporting acceptance testing.	D31
Other	Other hours spent on the project not covered above. Includes management, meetings, training hours, notebooks, system descriptions, user's guides, etc.	D32

SECTION C: Effort On Specific Activities (Need not add to A)
(Some hours may be counted in more than one area; view each activity separately)

Rework: Estimate of total hours spent that were caused by unplanned changes or errors. Includes effort caused by unplanned changes to specifications, erroneous or changed design, errors or unplanned changes to code, changes to documents. (This includes all hours spent debugging.) D33

Enhancing/Refining/Optimizing: Estimate of total hours spent improving the efficiency or clarity of design, or code, or documentation. These are not caused by required changes or errors in the system. D34

Documenting: Hours spent on any documentation of the system. Includes development of design documents, prologs, in-line commentary, test plans, system descriptions, user's guides, or any other system documentation. D35

Reuse: Hours spent in an effort to reuse components of the system. Includes effort in looking at other system(s) design, code, or documentation. Count total hours in searching, applying, and testing. D36

For Librarian's Use Only

Number: D37

Date: _____

Entered by: _____

Checked by: _____

10004437 g024

NOVEMBER 1991

Figure C-5. Personnel Resources Form (PRF)

Personnel Resources Form

(CLEANROOM VERSION)

Name: D21

Project: D1 Date (Friday): D22

SECTION A: Total Hours Spent on Project for the Week: _____

SECTION B: Hours By Activity (Total of hours in Section B should equal total hours in Section A)

Activity	Activity Definitions	Hours
Predesign	Understanding the concepts of the system. Any work prior to the actual design (such as requirements analysis).	D199
Pretest	Developing a test plan and building the test environment. Includes generating test cases, generating JCL, compiling components, building libranes, and defining inputs and probabilities.	D200
Create Design	Development of the system, subsystem, or components design. Includes box structure decomposition, stepwise refinement, development of PDL, design diagrams, etc.	D201
Verify/Review Design	Includes design meetings, formal and informal reviews, and walkthroughs.	D202
Write Code	Actually coding system components. Includes both desk and terminal code development.	D203
Read/Review Code	Code reading for any purpose other than isolation of errors. Includes verifying and reviewing code for correctness.	D204
Independent Test	Executing and evaluating tests of system components.	D205
Response to SFR	Isolating a tester-reported problem and developing a solution. Includes writing and reviewing design or code to isolate and correct a tester-reported problem.	D206
Acceptance Test	Running/supporting acceptance testing.	D207
Other	Other hours spent on the project not covered above. Includes management, meetings, training hours, notebooks, system descriptions, user's guides, etc.	D208

SECTION C: Effort On Specific Activities

Methodology Understanding/Discussion: Estimate the total hours spent learning, discussing, reviewing or attempting to understand cleanroom-related methods and techniques. Includes all time spent in training. D209

For Librarian's Use Only	
Number:	D210
Date:	
Entered by:	
Checked by:	

10004437 g025

NOVEMBER 1991

Figure C-6. Cleanroom Personnel Resources Form (CLPRF)

PROJECT COMPLETION STATISTICS FORM

Name: _____

Project: D1 Date: D2

Phase Dates (Saturdays)	
Phase	Start Date
Requirements Definition	D84
Design	D85
Implementation	D86
System Test	D87
Acceptance Test	D88
Cleanup	D89
Maintenance	D90
Project End	D91

Staff Resource Statistics	
Technical and Management Hours	D92
Services Hours	D93

Computer Resource Statistics		
Computer	CPU hours	No. of runs
D38	D94	D95

Project Size Statistics			
General Parameters		Source Lines of Code	
Number of subsystems	D96	Total	D100
Number of components	D97	New	D101
Number of changes	D98	Slightly Modified	D102
Pages of documentation	D99	Extensively Modified	D211
		Old	D103
		Comments	D104

Executable Modules		Executable Statements		Statements	
Total	D105	Total	D109	Total	D214
New	D106	New	D110	New	D215
Slightly Modified	D107	Slightly Modified	D111	Slightly Modified	D216
Extensively Modified	D212	Extensively Modified	D213	Extensively Modified	D217
Old	D108	Old	D112	Old	D218

Note: All of the values on this form are to be actual values at the completion of the project. The values entered by hand by SEL personnel reflect the data collected by the SEL during the course of the project. Update these according to project records and supply values for all blank fields.

For Librarian's Use Only

Number D113

Date: _____

Entered by: _____

Checked by _____

10004437 g026

NOVEMBER 1991

Figure C-7. Project Completion Statistics Form (PCSF)

PROJECT ESTIMATES FORM

Name: _____

Project: D1 _____ Date: D2 _____

Phase Dates (Saturdays)	
Phase	**Start Date**
Requirements Definition	D3
Design	D4
Implementation	D5
System Test	D6
Acceptance Test	D7
Cleanup	D8
Project End	D10

Staff Resource Estimates	
Programmer Hours	D11
Management Hours	D12
Services Hours	D13

Project Size Estimates	
Number of subsystems	D14
Number of components	D15
Source Lines of Code	
Total	D16
New	D17
Modified	D18
Old	D19

Note: All of the values on this form are to be estimates of projected values at completion of the project. This form should be submitted with updated estimates every 6 to 8 weeks during the course of the project.

For Librarian's Use Only
Number: D20
Date: _____
Entered by: _____
Checked by: _____

10004437-g027

NOVEMBER 1991

Figure C-8. Project Estimates Form (PEF)

PROJECT MESSAGES FORM

Name: _____

Project: D1 **Date:** D2

Messages:

P4, D61, D62

10004437-g028

NOVEMBER 1991

Figure C-9. Project Messages Form (PMF)

PROJECT STARTUP FORM

Name: _____

Project: D1 **Date:** D2

PLEASE PROVIDE ALL AVAILABLE INFORMATION

Project Full Name: P4, D61, D62

Project Type: P2, D60

Contacts: P4, D61, D62

Language: P4, D61, D62

Computer System: P4, D61, D62

Account: P4, D61, D62

Task Number: P4, D61, D62

Forms To Be Collected: (Circle forms that apply) P4, D61, D62

PEF PRF CLPRF DSF SPF SIF COF CCF CRF SEF PCSF WMEF MCRF

General Notes: P4, D61, D62

Personnel Names (indicate with * if not in database):

10004437-g029

NOVEMBER 1991

Figure C-10. Project Startup Form (PSF)

SERVICES/PRODUCTS FORM

Project: D1

Date (Friday): D22

COMPUTER RESOURCES

Computer	CPU Hours	No. of Runs
D38	D39	D40

GROWTH HISTORY

Components	D41
Changes	D42
Lines of Code	D43

SERVICES EFFORT

Service	Hours
Tech Pubs	D44
Secretary	D45
Proj Mgmt	D47
Other	D48

For Librarian's Use Only
Number: D49
Date:
Entered by:
Checked by:

10004437-g030

NOVEMBER 1991

Figure C-11. Services/Products Form (SPF)

SUBJECTIVE EVALUATION FORM

Name: _____

Project: D1 Date: D2

Indicate response by circling the corresponding numeric ranking.

I. PROBLEM CHARACTERISTICS

1. Assess the intrinsic difficulty or complexity of the problem that was addressed by the software development.

D114 1 (Easy) 2 3 (Average) 4 5 (Difficult)

2. How tight were schedule constraints on project?

D115 1 (Loose) 2 3 (Average) 4 5 (Tight)

3. How stable were requirements over development period?

D116 1 (Loose) 2 3 (Average) 4 5 (High)

4. Assess the overall quality of the requirements specification documents, including their clarity, accuracy, consistency, and completeness.

D117 1 (Low) 2 3 (Average) 4 5 (High)

5. How extensive were documentation requirements?

D118 1 (Low) 2 3 (Average) 4 5 (High)

6. How rigorous were formal review requirements?

D119 1 (Low) 2 3 (Average) 4 5 (High)

II. PERSONNEL CHARACTERISTICS: TECHNICAL STAFF

7. Assess overall quality and ability of development team.

D120 1 (Low) 2 3 (Average) 4 5 (High)

8. How would you characterize the development team's experience and familiarity with the application area of the project?

D121 1 (Low) 2 3 (Average) 4 5 (High)

9. Assess the development team's experience and familiarity with the development environment (hardware and support software).

D122 1 (Low) 2 3 (Average) 4 5 (High)

10. How stable was the composition of the development team over the duration of the project?

D123 1 (Loose) 2 3 (Average) 4 5 (High)

FOR LIBRARIAN'S USE ONLY

Number: _____ Entered by: _____

Date: D150 Checked by: _____

10004437 g031

NOVEMBER 1991

Figure C-12. Subjective Evaluation Form (SEF) (1 of 3)

SUBJECTIVE EVALUATION FORM

III. PERSONNEL CHARACTERISTICS: TECHNICAL MANAGEMENT

11. Assess the overall performance of project management.

D124 1 Low 2 3 Average 4 5 High

12. Assess project management's experience and familiarity with the application.

D125 1 Low 2 3 Average 4 5 High

13. How stable was project management during the project?

D126 1 Low 2 3 Average 4 5 High

14. What degree of disciplined project planning was used?

D127 1 Low 2 3 Average 4 5 High

15. To what degree were project plans followed?

D128 1 Low 2 3 Average 4 5 High

IV. PROCESS CHARACTERISTICS

16. To what extent did the development team use modern programming practices (PDL, top-down development, structured programming, and code reading)?

D129 1 Low 2 3 Average 4 5 High

17. To what extent did the development team use well-defined or disciplined procedures to record specification modifications, requirements questions and answers, and interface agreements?

D130 1 Low 2 3 Average 4 5 High

18. To what extent did the development team use a well-defined or disciplined requirements analysis methodology?

D131 1 Low 2 3 Average 4 5 High

19. To what extent did the development team use a well-defined or disciplined design methodology?

D132 1 Low 2 3 Average 4 5 High

20. To what extent did the development team use a well-defined or disciplined testing methodology?

D133 1 Low 2 3 Average 4 5 High

IV. PROCESS CHARACTERISTICS

21. What software tools were used by the development team? Check all that apply from the list that follows and identify any other tools that were used but are not listed.

D134

- ☐ Compiler
- ☐ Linker
- ☐ Editor
- ☐ Graphic display builder
- ☐ Requirements language processor
- ☐ Structured analysis support tool
- ☐ PDL processor
- ☐ ISPF
- ☐ SAP
- ☐ CAT
- ☐ PANVALET
- ☐ Test coverage tool
- ☐ Interface checker (RXVP80, etc.)
- ☐ Language-sensitive editor
- ☐ Symbolic debugger
- ☐ Configuration Management Tool (CMS, etc.)
- ☐ Others (identify by name and function)

22. To what extent did the development team prepare and follow test plans?

D135 1 Low 2 3 Average 4 5 High

10004437-g032

Figure C-12. Subjective Evaluation Form (SEF) (2 of 3)

SUBJECTIVE EVALUATION FORM

IV. PROCESS CHARACTERISTICS (CONT'D)

23. To what extent did the development team use well-defined and disciplined quality assurance procedures (reviews, inspections, and walkthroughs)?

D136 1 Low 2 3 Average 4 5 High

24. To what extent did development team use well-defined or disciplined configuration management procedures?

D137 1 Low 2 3 Average 4 5 High

V. ENVIRONMENT CHARACTERISTICS

25. How would you characterize the development team's degree of access to the development system?

D138 1 Low 2 3 Average 4 5 High

26. What was the ratio of programmers to terminals?

D139 1 8:1 2 4:1 3 2:1 4 1:1 5 1:2

27. To what degree was the development team constrained by the size of main memory or direct-access storage available on the development system?

D140 1 Low 2 3 Average 4 5 High

28. Assess the system response time: were the turnaround times experienced by the team satisfactory in light of the size and nature of the jobs?

D141 1 Poor 2 3 Average 4 5 Very Good

29. How stable was the hardware and system support software (including language processors) during the project?

D142 1 Low 2 3 Average 4 5 High

30. Assess the effectiveness of the software tools.

D143 1 Low 2 3 Average 4 5 High

VI. PRODUCT CHARACTERISTICS

31. To what degree does the delivered software provide the capabilities specified in the requirements?

D144 1 Low 2 3 Average 4 5 High

32. Assess the quality of the delivered software product.

D145 1 Low 2 3 Average 4 5 High

33. Assess the quality of the design that is present in the software product.

D146 1 Low 2 3 Average 4 5 High

34. Assess the quality and completeness of the delivered system documentation.

D147 1 Low 2 3 Average 4 5 High

35. To what degree were software products delivered on time?

D148 1 Low 2 3 Average 4 5 High

36. Assess smoothness or relative ease of acceptance testing.

D149 1 Low 2 3 Average 4 5 High

10004437-g033

Figure C-12. Subjective Evaluation Form (SEF) (3 of 3)

SUBSYSTEM INFORMATION FORM

Name: _____

Project: D1 Date: D2

Add New Subsystems

Subsystem Prefix	Subsystem Name	Subsystem Function
D50	D51	D52

Change Existing Subsystems

Old Subsystem Prefix (Must exist in the database)	Action (R - Rename, D - Delete)	New Subsystem Prefix (Must not exist in the database)

This form is to be completed by the time of the Preliminary Design Review (PDR). An update must be submitted each time a new subsystem is defined thereafter. This form is also to be used when a subsystem is renamed or deleted.

Subsystem Prefix: A prefix of 2 to 5 characters used to identify the subsystem when naming components

Subsystem Name: A descriptive name of up to 40 characters

Subsystem Function: Enter the most appropriate function code from the list of functions below:

USERINT:	User Interface
DPDC:	Data Processing/Data Conversion
REALTIME:	Real-time Control
MATHCOMP:	Mathematical/Computational
GRAPH:	Graphics and Special Device Support
CPEXEC:	Control Processing/Executive
SYSSERV:	System Services

For Librarian's Use Only
Number: _____
Date: _____
Entered by: _____
Checked by: _____

10004437 g034

NOVEMBER 1991

Figure C-13. Subsystem Information Form (SIF)

WEEKLY MAINTENANCE EFFORT FORM

Name: D21

Project: D1 Date (Friday): D22

For Librarian's Use Only

Number: D161

Date:

Entered by:

Checked by:

Section A – Total Hours Spent on Maintenance (Includes time spent on all maintenance activities for the project excluding writing specification modifications)

Section B – Hours By Class of Maintenance (Total of hours in Section B should equal total hours in Section A)

Class	Definition	Hours
Correction	Hours spent on all maintenance associated with a system failure.	D151
Enhancement	Hours spent on all maintenance associated with modifying the system due to a requirements change. Includes adding, deleting, or modifying system features as a result of a requirements change.	D152
Adaptation	Hours spent on all maintenance associated with modifying a system to adapt to a change in hardware, system software, or environmental characteristics.	D153
Other	Other hours spent on the project (related to maintenance) not covered above. Includes management, meetings, etc.	D154

Section C – Hours By Maintenance Activity (Total of hours in Section C should equal total hours in Section A)

Activity	Activity Definitions	Hours
Isolation	Hours spent understanding the failure or request for enhancement or adaptation.	D155
Change Design	Hours spent actually redesigning the system based on an understanding of the necessary change.	D156
Implementation	Hours spent changing the system to complete the necessary change. This includes changing not only the code, but the associated documentation.	D157
Unit Test/ System Test	Hours spent testing the changed or added components. Includes hours spent testing the integration of the components.	D158
Acceptance/ Benchmark Test	Hours spent acceptance testing or benchmark testing the modified system.	D159
Other	Other hours spent on the project (related to maintenance) not covered above. Includes management, meetings, etc.	D160

10004437-g035

NOVEMBER 1991

Figure C-14. Weekly Maintenance Effort Form (WMEF)

APPENDIX D—DATA DEFINITION LANGUAGE FOR THE SEL DATABASE

This appendix describes the data definition language (DDL) that contains all the semantic rules of the SEL database. This DDL represents the design of the SEL database. It is not implementation language and should not be confused with Oracle's DDL statements in SQL.

In the design DDL, each base relation is identified by the keyword RELATION and each view is identified by the keyword VIEW. Each field within a relation is identified by the keyword FIELD followed by its name, its data type, and its length. Char, which represents a character data type, is followed by the maximum length of the field. Numeric, which represents a numeric data type, is followed by the width of the field and the number of decimal places, if any. Date represents an ORACLE date data type.

The primary key component(s) is identified by the keyword KEY. The keyword UNIQUE identifies fields that are not part of the primary key but whose values are unique within a relation. The keyword INDEX identifies fields that are not unique, but should be indexed to facilitate database retrievals.

The constraints mentioned in Section 4.2.3 are represented by mathematical expressions. The following constraint in the DDL

CONSTRAINT

RANGE PROJECT P

RANGE PROJ_SUB S

$$\forall S \ \exists P \ (P.PROJ\_NO = S.PROJ\_NO)$$

can be interpreted as follows: P is the range variable that ranges over the PROJECT relation, and its permitted values are records of PROJECT. S is the range variable that ranges over the PROJ_SUB relation, and its permitted values are records of PROJ_SUB. Here, range variables are used as a simple shorthand. For all ($\forall$) S, there exists ($\exists$) P such that PROJ_NO in P is equal to PROJ_NO in S. In other words, for each project number that exists in the project–subsystem relation, the same project number must exist in the project relation. Besides "for all" ($\forall$) and "there exist" ($\exists$) qualifiers, the qualifier "or" ($\vee$) is used in the constraint definition of relation EFF_ACT, and the qualifier "and" $\wedge$ is used in the constraint definitions of relations CH_ERR_ARES, CH_ERR_TOOLS, CH_ADAFEAT, and CH_ERR_GEN. Each field within a view is identified by the keyword FIELD followed by its name and the base relation from which it is derived. The field lengths are the same as in the base relations.

RELATION CHANGE

(FIELD CHANGE_NO char (6)
FIELD PROG_ID numeric(5)
FIELD SUB_DATE date
FIELD EFF_ONE char(1)
FIELD EFF_ADA char(1)
FIELD EFF_ISO_CH char(10)
FIELD EFF_COM_CH char(10)
FIELD EFF_PARPA char(1)
FIELD EFF_OTHER char(1)
FIELD DATE_DETER date
FIELD DATE_COMP date
FIELD NUM_COM_CH numeric(2)
FIELD NUM_COM_EX numeric(2)
FIELD CH_TYPE char(10)
FIELD FORM_TYPE char(6)
FIELD STATUS char(10))

KEY (CHANGE_NO)
INDEX (SUB_DATE)
INDEX (PROG_ID)
INDEX (CH_TYPE)
INDEX (STATUS)
CONSTRAINT

RANGE VAL_ISO_CH VEI
RANGE CHANGE CH
RANGE PERSONNEL PROG
RANGE VAL_STATUS VS
RANGE VAL_EFF_COM_CH VEC
RANGE VAL_CH_TYPE VCHT

∀CH ∃PROG (PROG.PROG_ID = CH.PROG_ID)
∀CH ∃VS (VS.CODE = CH.STATUS)
∀CH ∃VEI (VEI.CODE = CH.EFF_ISO_CH)
∀CH ∃VEC (VEC.CODE = CH.EFF_COM_CH)
∀CH ∃VCHT (VCHT.CODE = CH.CH_TYPE)
∀CH ∃CH (CH.FORM_TYPE = 'CRF')

```
RELATION CHANGE_COM
        (FIELD CHANGE_NO char(6)
        FIELD COM_NO numeric(7))
    KEY (CHANGE_NO, COM_NO)
    INDEX (COM_NO)
    CONSTRAINT
        RANGE SUB_COM C
        RANGE CHANGE_COM CHC
        RANGE CHANGE CH
                ∀CHC   ∃C (C.COM_NO = CHC.COM_NO)
                ∀CHC   ∃CH (CH.CHANGE_NO = CHC.CHANGE_NO)

RELATION CH_ADAFEAT
        (FIELD CHANGE_NO char(6)
        FIELD ADA_FEATURE char(10))
    KEY (CHANGE_NO, ADA_FEATURE)
    CONSTRAINT
        RANGE CHANGE CH
        RANGE CH_ADAFEAT CHA
        RANGE VAL_ADA_FEATURE VAF
                ∀CHA   ∃VAF (VAF.CODE = CHA.ADA_FEATURE)
                ∀CHA   ∃CH (CH.EFF_ADA = 'Y' ∧ CH.CHANGE_NO =
                                CHA.CHANGE_NO ∧ CH.CH_TYPE = 'ERRCO')

RELATION CH_ERR_ARES
        (FIELD CHANGE_NO char(6)
        FIELD ERR_ARES char(10))
    KEY (CHANGE_NO, ERR_ARES)
    CONSTRAINT
        RANGE CHANGE CH
        RANGE CH_ERR_ARES CHEA
        RANGE VAL_ERR_ARES VEA
                ∀CHEA   ∃CH (CH.CH_TYPE = 'ERRCO' ∧ CH.CHANGE_NO =
                                CHEA.CHANGE_NO ∧ CH.EFF_ADA = 'Y')
                ∀CHEA   ∃VEA (VEA.CODE = CHEA.ERR_ARES)
```

```
RELATION CH_ERR_GEN
        (FIELD CHANGE_NO char(6)
        FIELD ERR_SOURCE char(10)
        FIELD ERR_CLASS char(10)
        FIELD ERR_COMIS char(1)
        FIELD ERR_TYPO char(1)
        FIELD ERR_OMIS char(1)
        FIELD ERR_ADOC char(1)
        FIELD ERR_ACAUSE char(10))
    KEY (CHANGE_NO)
    INDEX (ERR_ACAUSE)
    CONSTRAINT
        RANGE CHANGE CH
        RANGE CH_ERR_GEN CHEG
        RANGE VAL_ERR_SOURCE VES
        RANGE VAL_ERR_CLASS VEC
        RANGE VAL_ERR_ACAUSE VERA
              ∀CHEG  ∃CH (CH.CH_TYPE = 'ERRCO' ∧ CH.CHANGE_NO =
                          CHEG.CHANGE_NO)
              ∀CHEG  ∃VES (VES.CODE = CHEG.ERR_SOURCE)
              ∀CHEG  ∃VERA (VERA.CODE = CHEG.ERR_ACAUSE)
              ∀CHEG  ∃VEC (VEC.CODE = CHEG.ERR_CLASS)

RELATION CH_ERR_TOOLS
        (FIELD CHANGE_NO char(6)
        FIELD ERR_TOOLS char(10))
    KEY (CHANGE_NO, ERR_TOOLS)
    CONSTRAINT
        RANGE CHANGE CH
        RANGE CH_ERR_TOOLS CHET
        RANGE VAL_ERR_TOOLS VET
              ∀CHET  ∃CH (CH.CH_TYPE = 'ERRCO' ∧ CH.CHANGE_NO =
                          CHET.CHANGE_NO)
              ∀CHET  ∃VET (VET.CODE = CHET.ERR_TOOLS)
```

```
RELATION COMPUTER
        (FIELD CPU_NAME char(10)
        FIELD C_FULL_NAME char(20))
        KEY (CPU_NAME)

RELATION COM_PURPOSE
        (FIELD COM_NO numeric(7)
        FIELD PURPOSE char(10))
    KEY (COM_NO, PURPOSE)
    CONSTRAINT
        RANGE COM_SOURCE C
        RANGE COM_PURPOSE CP
        RANGE VAL_COM_PURPOSE_VCOP
                ∀CP  ∃C (C.COM_NO = CP.COM_NO)
                ∀CP  ∃VCOP (VCOP.CODE = CP.PURPOSE)

RELATION COM_SOURCE
        (FIELD COM_NO numeric(7)
        FIELD PROG_ID numeric(5)
        FIELD FORM_NO char(6)
        FIELD FORM_TYPE char(6)
        FIELD STATUS char(10)
        FIELD CREATE_DATE date
        FIELD ORI_TYPE char(10)
        FIELD COM_TYPE char(10)
        FIELD DIFFICULTY numeric(2)
        FIELD SUB_DATE date)
    KEY (COM_NO)
    UNIQUE (FORM_NO)
    INDEX (FORM_NO)
    INDEX (STATUS)
    INDEX (CREATE_DATE)
    INDEX (SUB_DATE)
    CONSTRAINT
        RANGE SUB_COM C
        RANGE COM_SOURCE CSO
```

```
        RANGE VAL_ORI_TYPE VOT
        RANGE VAL_STATUS VS
        RANGE VAL_COM_TYPE VCT
        RANGE PERSONNEL PROG
                ∀CSO  ∃C (C.COM_NO = CSO.COM_NO)
                ∀CSO  ∃VOT (VOT.CODE = CSO.ORI_TYPE)
                ∀CSO  ∃VS (VS.CODE = CSO.STATUS)
                ∀CSO  ∃VCT (VCT.CODE = CSO.COM_TYPE)
                ∀CSO  ∃PROG (PROG.PROG_ID = CSO.PROG_ID)
                ∀CSO  ∃CSO (CSO.FORM_TYPE = 'COF')

RELATION COM_STAT
        (FIELD COM_NO numeric(7)
        FIELD C_EXE_S numeric(6)
        FIELD C_LINE numeric(6)
        FIELD C_C_LINE numeric(6)
        FIELD C_STMTS numeric(6)
        FIELD FINAL_ORIGIN_CAT char(10))
    KEY (COM_NO)
    CONSTRAINT
        RANGE SUB_COM C
        RANGE COM_STAT CS
                ∀CS  ∃C (C.COM_NO = CS.COM_NO)

RELATION CRF_TEMP_CHANGE_COM
        (FIELD USER_ID numeric
        FIELD SUB_PRE char(5)
        FIELD COM_NAME char(40)
        FIELD COM_NO numeric(7))
    KEY (USER_ID, SUB_PRE, COM_NAME)
    CONSTRAINT
        RANGE V_PROJ_COM VPROJ
        RANGE CRF_TEMP_CHANGE_COM CRF
        RANGE PROJ_SUB SUB
                ∀CRF  ∃SUB (SUB.SUB_PRE = CRF.SUB_PRE)
                ∀CRF  ∃VPROJ (VPROJ.COM_NAME = CRF.COM_NAME)
                ∀CRF  ∃VPROJ (VPROJ.COM_NO = CRF.COM_NO)
```

```
RELATION DSF_MEASURE
        (FIELD D_ID numeric(10)
        FIELD STATUS_CODE char(10)
        FIELD MEASURE_CODE char(10)
        FIELD MEASURE_VALUE numeric(5))
    KEY (D_ID, STATUS_CODE, MEASURE_CODE)
    CONSTRAINT
        RANGE VAL_DSF_TARGET VDT
        RANGE VAL_DSF_MEASURE VDM
        RANGE PROJ_DSF DSF
        RANGE DSF_MEASURE DM
                ∀DM  ∃VDT (VDT.CODE = DM.MEASURE_CODE)
                ∀DM  ∃VDM (VDM.CODE = DM.STATUS_CODE)
                ∀DM  ∃DSF (DSF.D_ID = DM.D_ID)

RELATION DSF_TARGET
        (FIELD D_ID numeric(10)
        FIELD STATUS_CODE char(10)
        FIELD TARGET_CODE char(10)
        FIELD TARGET_VALUE numeric(5))
    KEY (D_ID, STATUS_CODE, TARGET_CODE)
    CONSTRAINT
        RANGE VAL_DSF_TARGET VDT
        RANGE VAL_DSF_STATUS VDS
        RANGE PROJ_DSF DSF
        RANGE DSF_TARGET DT
                ∀DT  ∃VDT (VDT.CODE = DT.TARGET_CODE)
                ∀DT  ∃VDS (VDS.CODE = DT.STATUS_CODE)
                ∀DT  ∃DSF (DSF.D_ID = DT.D_ID)

RELATION DUMMY
        (FIELD HIDDEN char(1))

RELATION EFF_ACT
        (FIELD EFF_ID numeric(10)
        FIELD ACTIVITY char(10)
        FIELD ACT_HR numeric(10, 2))
```

```
    KEY (EFF_ID, ACTIVITY)
    CONSTRAINT
        RANGE EFF_PROJ EP
        RANGE EFF_SUB ES
        RANGE VAL_ACTIVITY VA
        RANGE EFF_ACT EA
                ∀EA   ∃VA (VA.CODE = EA.ACTIVITY)
                ∀EA   ∃EP ES (ES.PS_ID = EA.EFF_ID
                              EP.P_ID = EA.EFF_ID)

RELATION EFF_FORM
        (FIELD P_ID numeric(10)
        FIELD FORM_NO char(6)
        FIELD FORM_TYPE char(6)
        FIELD STATUS char(10))
    KEY (P_ID)
    INDEX (STATUS)
    INDEX (FORM_NO)
    CONSTRAINT
        RANGE EFF_PROJ EP
        RANGE EFF_FORM EFF
        RANGE VAL_STATUS VS
                ∀EFF   ∃EP (EP.P_ID = EFF.P_ID)
                ∀EFF   ∃VS (VS.CODE = EFF.STATUS)
                ∀EFF   ∃EFF (EFF.FORM_TYPE = 'SPF' V EFF.FORM_TYPE = 'PRF')

RELATION EFF_PROJ
        (FIELD PROJ_NO numeric(3)
        FIELD SUB_DATE date
        FIELD PROG_ID numeric(5)
        FIELD P_ID numeric(10))
    KEY (PROJ_NO, SUB_DATE, PROG_ID)
    UNIQUE (P_ID)
    INDEX (P_ID)
    CONSTRAINT
        RANGE PROJECT P
        RANGE PERSONNEL PROG
        RANGE EFF_PROJ EP
```

∀EP ∃P (P.PROJ_NO = EP.PROJ_NO)

∀EP ∃PROG (PROG.PROG_ID = EP.PROG_ID)

∀EP ∃EP (EP.SUB_DATE = a valid Friday date)

RELATION EFF_SUB

(FIELD P_ID numeric(10)

FIELD SUB_PRE char(5)

FIELD PS_ID numeric(10))

KEY (P_ID, SUB_PRE)

UNIQUE (PS_ID)

INDEX (PS_ID)

CONSTRAINT

RANGE EFF_PROJ EP

RANGE EFF_SUB ES

RANGE PROJ_SUB S

∀ES ∃S (S.SUB_PRE = ES.SUB_PRE)

∀ES ∃EP (EP.P_ID = ES.P_ID)

RELATION GENERATE_SAT_DAY

(FIELD SCRIPT_NO numeric(10)

FIELD SAT_DAY date)

KEY (SCRIPT_NO, SAT_DAY)

CONSTRAINT

RANGE TEMP_SCRIPT T

RANGE GENERATE_SAT_DAY SAT

∀SAT ∃T (T.SCRIPT_NO = SAT.SCRIPT_NO)

∀SAT ∃SAT (SAT.SAT_DAY = a valid Saturday date)

RELATION MAINT_ACT_HRS

(FIELD MAINT_ID numeric(10)

FIELD MAINT_ACT char(10)

FIELD ACT_HR numeric(10, 2))

KEY (MAINT_ID, MAINT_ACT)

```
    CONSTRAINT
        RANGE MAINT_ACT_HRS MAH
        RANGE MAINT_PROF MP
        RANGE VAL_ACT VA
                ∀MAH  ∃VC (VA.CODE = MAH.MAINT_ACT)
                ∀MAH  ∃MP (MP.MAINT_ID = MAH.MAINT_ID)

RELATION MAINT_CHANGE
        (FIELD MAINT_CH_NO char(6)
        FIELD PROJ_NO numeric(3)
        FIELD PROG_ID numeric(5)
        FIELD SUB_DATE date
        FIELD OSMR_NO numeric(4)
        FIELD STATUS char(10)
        FIELD FORM_TYPE char(6)
        FIELD MAINT_CH_TYPE char(10)
        FIELD CH_CAUSE char(10)
        FIELD MAINT_ISO_CH char(10)
        FIELD MAINT_COM_CH char(10)
        FIELD CH_CLASS char(10)
        FIELD EST_LOC_ADD numeric(6)
        FIELD EST_LOC_CH numeric(6)
        FIELD EST_LOC-DEL numeric(6)
        FIELD COMP_ADD numeric(4)
        FIELD COMP_CH numeric(4)
        FIELD COMP_DEL numeric(4)
        FIELD COMP_ADD_NEW numeric(4)
        FIELD COMP_ADD_REUSE numeric(4)
        FIELD COMP_ADD_REMOD numeric(4)
    KEY (MAINT_CH_NO)
    INDEX (PROJ_NO)
    CONSTRAINT
        RANGE MAINT_CHANGE MC
        RANGE VAL_MAINT_CH_TYPE VMCT
        RANGE VAL_CH CAUSE VCHC
        RANGE PROJECT P
```

```
        RANGE VAL_STATUS VS
        RANGE PERSONNEL PROG
        RANGE VAL_MAINT_ISO_CH VMIC
        RANGE VAL_MAINT_COM_CH VMCC
        RANGE VAL_CH_CLASS VCC
                ∀MC   ∃P (P.PROJ_NO = MC.PROJ_NO)
                ∀MC   ∃PROG (PROG.PROG_ID = MC.PROG_ID)
                ∀MC   ∃VS (VS.CODE = MC.STATUS)
                ∀MC   ∃MC (MC.FORM_TYPE = 'MCRF')
                ∀MC   ∃VMCT (VMCT.CODE = MC.MAINT_CH_TYPE)
                ∀MC   ∃VCHC (VCHC.CODE = MC.CH_CAUSE)
                ∀MC   ∃VMIC (VMIC.CODE = MC.MAINT_ISO_CH)
                ∀MC   ∃VMCC (VMCC.CODE = MC.MAINT_COM_CH)
                ∀MC   ∃VCC (VCC.CODE = MC.CH_CLASS)
                ∀MC   ∃MC (SUM(MC.COMP_ADD) =
                                SUM(MC.COMP_ADD_NEW+
                                MC.COMP_ADD_REUSE+
                                MC.COMP_ADD_REMOD))

RELATION MAINT_CH_OBJECTS
        (FIELD MAINT_CH_NO char(6)
        FIELD CH_OBJECT char(10))
    KEY (MAINT_CH_NO, CH_OBJECT)
    CONSTRAINT
        RANGE MAINT_CH_OBJECTS MCO
        RANGE VAL_CH_OBJECT VCO
        RANGE MAINT_CHANGE MC
                ∀MCO   ∃VCO (VCO.CODE = MCO.CH_OBJECT)
                ∀MCO   ∃MC (MC.MAINT_CH_NO = MCO.MAINT_CH_NO)

RELATION MAINT_CLASS_HRS
        (FIELD MAINT_ID numeric(10)
        FIELD MAINT_CLASS char(10)
        FIELD CLASS_HR numeric(10, 2))
    KEY (MAINT_ID, MAINT_CLASS)
```

```
    CONSTRAINT
        RANGE MAINT_CLASS_HRS MCH
        RANGE MAINT_PROJ MP
        RANGE VAL_CLASS VC
                ∀MCH   ∃VC (VC.CODE = MCH.MAINT_CLASS)
                ∀MCH   ∃MP (MP.MAINT_ID = MCH.MAINT_ID)

RELATION MAINT_PROJ
        (FIELD PROJ_NO numeric(3)
        FIELD SUB_DATE date
        FIELD PROG_ID numeric(5)
        FIELD MAINT_ID numeric(10)
        FIELD FORM_NO char(6)
        FIELD FORM_TYPE char(6)
        FIELD STATUS char(10))
    KEY (PROJ_NO, SUB_DATE, PROG_ID)
    UNIQUE (MAINT_ID)
    INDEX (MAINT_ID)
    INDEX (FORM_NO)
    CONSTRAINT
        RANGE MAINT_PROJ MP
        RANGE PROJECT P
        RANGE VAL_STATUS VS
        RANGE PERSONNEL PROG
                ∀MP   ∃P (P.PROJ_NO = MP.PROJ_NO)
                ∀MP   ∃PROG (PROG.PROG_ID = MP.PROG_ID)
                ∀MP   ∃VS (VS.CODE = MP.STATUS)
                ∀MP   ∃MP (MP.SUB_DATE – a valid Friday date)
                ∀MP   ∃MP (MP.FORM_TYPE = 'WMEF')

RELATION PC_SEQNO
        (FIELD TABLE_NAME char(30)
        FIELD FIELD_NAME char(30)
        FIELD MAX_SEQNO numeric(10))
    KEY (TABLE_NAME, FIELD_NAME)
```

```
    CONSTRAINT
        RANGE PC SEQNO S
                ∀S   ∃S  (S.TABLE_NAME = a valid relation name
                          ∧ S.FIELD_NAME = a valid field name within that
                          relation)

RELATION PERM_SCRIPT
        (FIELD ORA_USER char(20)
        FIELD SCRIPT_NAME char(20)
        FIELD SCRIPT_NO numeric(10))
        FIELD OUT_ROUTING char(20)
        FIELD OUT_FILE char(20)
    KEY (ORA_USER, SCRIPT_NAME)
    UNIQUE (SCRIPT_NO)
    INDEX (SCRIPT_NO)
    CONSTRAINT
        RANGE USER_CLASS U
        RANGE PERM_SCRIPT P
                ∀P   ∃U  (U.ORA_USER = P.ORA_USER)
                ∀P   ∃P  ((P.OUT_ROUTING = '2')
                          ∧ (P.OUT_FILE != null ∧
                            P.OUT_ROUTING = '1'))

RELATION PERSONNEL
        (FIELD PROG_ID numeric(5)
        FIELD FORM_NAME char(15)
        FIELD FULL_NAME char(30)
        FIELD DATE_ENTRY date)
    KEY (PROG_ID)
    UNIQUE (FORM_NAME)
    INDEX (FORM_NAME)

RELATION PROJECT
        (FIELD PROJ_NAME char(8)
        FIELD PROJ_NO numeric(3)
        FIELD PROJ_TYPE char(10))
        (FIELD ACTIVE_STATUS char(10))
    KEY (PROJ_NAME)
    UNIQUE (PROJ_NO)
    INDEX (PROJ_NO)
```

```
RELATION PROJ_CPU_STAT
        (FIELD PROJ_NO numeric(3)
        FIELD SUB_DATE date
        FIELD CPU_NAME char(10)
        FIELD TOTAL_HRS numeric(10,2)
        FIELD T_RUN numeric(6))
    KEY (PROJ_NO, SUB_DATE, CPU_NAME)
    CONSTRAINT
        RANGE PROJECT P
        RANGE PROJ_EST_CPU PESC
        RANGE COMPUTER_CPU
                ∀PESC   ∃P (P.PROJ_NO = PESC.PROJ_NO)
                ∀PESC   ∃CPU (CPU.CPU_NAME = PESC.CPU_NAME)

RELATION PROJ_DSF
        (FIELD PROJ_NO numeric(3)
        FIELD SUB_DATE date
        FIELD PROG_ID numeric(5)
        FIELD FORM_NO char(6)
        FIELD STATUS char(10)
        FIELD FORM_TYPE char(6)
        FIELD D_ID numeric(10))
    KEY (PROJ_NO, SUB_DATE)
    UNIQUE (D_ID)
    UNIQUE (FORM_NO)
    INDEX (D_ID)
    INDEX (FORM_NO)
    CONSTRAINT
        RANGE VAL_STATUS VDS
        RANGE PERSONNEL PROG
        RANGE PROJECT P
        RANGE PROJ_DSF_PD
                ∀PD   ∃P (P.PROJ_NO = PD. PROJ_NO)
                ∀PD   ∃PROG (PROG.PROG_ID = PD.PROG_ID)
                ∀PD   ∃PD (PD.SUB_DATE = a valid Friday date)
                ∀PD   ∃VDS (VDS.CODE = PD.STATUS)
                ∀PD   ∃PD (PD.FORM_TYPE = 'DSF')
```

```
RELATION PROJ_EST
        (FIELD PROJ_NO numeric(3)
        FIELD SUB_DATE date
        FIELD T_SYS numeric(4)
        FIELD T_COM numeric(4)
        FIELD T_LINE numeric(7)
        FIELD T_NEW_LINE numeric(7)
        FIELD T_MOD_LINE numeric(7)
        FIELD T_OLD_LINE numeric(7)
        FIELD PRO_HR numeric(10,2)
        FIELD MAN_HR numeric(10,2)
        FIELD SER_HR numeric(10,2)
    KEY (PROJ_NO, SUB_DATE)
    CONSTRAINT
        RANGE PROJECT P
        RANGE PROJ_EST PES
                ∀PES   ∃P (P.PROJ_NO = PES.PROJ_NO)

RELATION PROJ_EST_PHASE
        (FIELD PROJ_NO numeric(3)
        FIELD SUB_DATE date
        FIELD PHASE_CO char(10)
        FIELD START_DATE date
        FIELD END_DATE date)
    KEY (PROJ_NO, SUB_DATE, PHASE_CO)
    CONSTRAINT
        RANGE PROJECT P
        RANGE PROJ_EST_PHASE PESP
        RANGE VAL_PHASE_CO VPC
                ∀PESP   ∃P (P.PROJ_NO = PESP.PROJ_NO)
                ∀PESP   ∃VPC (VPC.CODE = PESP.PHASE_CO)
                ∀PESP   ∃PESP (PESP.START_DATE = a valid Saturday date)
                ∀PESP   ∃PESP (PESP.END_DATE = a valid Saturday date)
```

```
RELATION PROJ_FORM
        (FIELD PROJ_NO numeric(3)
        FIELD SUB_DATE date
        FIELD FORM_NO char(6)
        FIELD FORM_TYPE char(6)
        FIELD STATUS char(10))
    KEY (PROJ_NO, SUB_DATE, FORM_TYPE)
    UNIQUE (FORM_NO, FORM_TYPE)
    INDEX (FORM_TYPE)
    INDEX (STATUS)
    CONSTRAINT
        RANGE PROJECT P
        RANGE PROJ_FORM PF
        RANGE VAL_STATUS VS
                ∀PF   ∃P (P.PROJ_NO = PF.PROJ_NO)
                ∀PF   ∃VS (VS.COD = PF.STATUS)
                ∀PF   ∃PF (PF.FORM_TYPE = 'PEF' V PF.FORM_TYPE =
                        'SPF' V PF.FORM_TYPE = 'PCSF' V
                        PF.FORM_TYPE = 'SEF')

RELATION PROJ_GRH
        (FIELD PROJ_NO numeric(3)
        FIELD SUB_DATE date
        FIELD GR_LINE numeric(7)
        FIELD GR_MOD numeric(4)
        FIELD GR_CH numeric(6))
    KEY (PROJ_NO, SUB_DATE)
    CONSTRAINT
        RANGE PROJECT P
        RANGE PROJ_GRH PG
                ∀PG   ∃P (P.PROJ_NO = PG.PROJ_NO)
                ∀PG   ∃PG (PG.SUB_DATE = a valid Friday date)

RELATION PROJ_MESSAGES
        (FIELD S_ID numeric(5)
        FIELD LINE_NO numeric (3)
        FIELD MESSAGES char (65)
        FIELD SUB_DATE date)
```

KEY (S_ID, LINE_NO)
CONSTRAINT
RANGE PROJ_NOTES PN
RANGE PROJ_MESSAGES PM
∀PN ∃PM (PM.S_ID = PN.S_ID)

RELATION PROJ_NOTES
(FIELD PROJ_NO numeric(3)
FIELD NOTE_TYPE char(10)
FIELD S_ID numeric(5))
KEY (PROJ_NO, NOTE_TYPE)
UNIQUE (S_ID)
INDEX (S_ID)
CONSTRAINT
RANGE PROJECT P
RANGE VAL_NOTE_TYPE VNT
RANGE PROJ_NOTES PN
∀PN ∃P (P.PROJ_NO = PN.PROJ_NO)
∀PN ∃VNT_(VNT.CODE = PN.NOTE_TYPE)

RELATION PROJ_PROD
(FIELD PROJ_NO numeric(3)
FIELD SUB_DATE date
FIELD RES_NAME char(10)
FIELD RES_HR numeric(10,2)
FIELD RES_RUN numeric(5))
KEY (PROJ_NO, SUB_DATE, RES_NAME)
CONSTRAINT
RANGE PROJECT P
RANGE PROJ_PROD PR
RANGE COMPUTER CPU
∀PR ∃P (P.PROJ_NO = PR.PROJ_NO)
∀PR ∃CPU (CPU.CPU_NAME = PR.RES_NAME)
∀PR ∃PR (PR.SUB_DATE = a valid Friday date)

RELATION PROJ_SEF

(FIELD PROJ_NO numeric(3)

FIELD MEAS_TYPE char(10)

FIELD EVALUATE numeric(1))

KEY (PROJ_NO, MEAS_TYPE)

CONSTRAINT

RANGE PROJECT P

RANGE PROJ_SEF PSE

RANGE VAL_MEAS_TYPE VMT

∀PSE ∃P (P.PROJ_NO = PSE.PROJ_NO)

∀PSE ∃VMT_(VMT.CODE = PSE.MEAS_TYPE)

RELATION PROJ_SEF_SEC

(FIELD PROJ_NO numeric(3)

FIELD MEAS_TYPE char(10)

FIELD SECOND_L char(10))

KEY (PROJ_NO, MEAS_TYPE, SECOND_L)

CONSTRAINT

RANGE PROJ_SEF_SEC PSES

RANGE PROJ_SEF PSE

RANGE VAL_SEC_L VSL

∀PSES ∃PSE (PSE.MEAS_TYPE = PSES.MEAS_TYPE ∧
PSE.PROJ_NO = PSES.PROJ_NO)

∀PSES ∃VSL (VSL.CODE = PSES.SECOND L)

RELATION PROJ_STAT

(FIELD PROJ_NO numeric(3)

FIELD SUB_DATE date

FIELD TECH_MAN_HR numeric(10,2)

FIELD SER_HR numeric(10,2)

FIELD T_SYS numeric(4)

FIELD T_COM numeric(4)

FIELD T_CH numeric(6)

FIELD T_DOC numeric(6)

FIELD T_LINE numeric(7)

```
        FIELD T_NEW_LINE numeric(6)
        FIELD T_MOD_LINE numeric(6)
        FIELD T_OLD_LINE numeric(6)
        FIELD T_COMMENT numeric(6)
        FIELD T_EXE_MOD numeric(4)
        FIELD T_NEW_MOD numeric(4)
        FIELD T_MOD_MOD numeric(4)
        FIELD T_OLD_MOD numeric(4)
        FIELD T_EXE_STAT numeric(6)
        FIELD T_NEW_STAT numeric(6)
        FIELD T_MOD_STAT numeric(6)
        FIELD T_OLD_STAT numeric(6)
        FIELD T_STMTS numeric(6)
        FIELD T_NEW_STMTS numeric(6)
        FIELD T_MOD_STMTS numeric(6)
        FIELD T_OLD_STMTS numeric(6))
        FIELD T_EXTMO_LINE numeric(6)
        FIELD T_EXTMO_MOD numeric(4)
        FIELD T_EXTMO_STAT numeric(6)
        FIELD T_EXTMO_STMTS numeric(6))
    KEY (PROJ_NO)
    CONSTRAINT
        RANGE PROJECT P
        RANGE PROJ_EST PES
                ∀PES  ∃P (P.PROJ_NO = PES.PROJ_NO)

RELATION PROJ_SUB
        (FIELD PROJ_NO numeric(3)
        FIELD SUB_PRE char(5)
        FIELD SUB_DATE date
        FIELD SUBSY_ID numeric(5))
    KEY (PROJ_NO, SUB_PRE)
    UNIQUE (SUBSY_ID)
    INDEX (SUBSY_ID)
```

```
    CONSTRAINT
        RANGE PROJECT P
        RANGE PROJ_SUB S
              ∀S  ∃P (P.PROJ_NO = S.PROJ_NO)

RELATION REP_CODES
        (FIELD CODE char(10)
        FIELD VALUE char(30)
        FIELD FUNCTION char(15))
    KEY (CODE)

RELATION SCRIPT_PROJECTS
        (FIELD SCRIPT_NO numeric(10)
        FIELD REPORT_SEQ numeric(3)
        FIELD PROJ_NAME char(8))
    KEY (SCRIPT_NO, REPORT_SEQ, PROJ_NAME)
    CONSTRAINT
        RANGE PROJECT PR
        RANGE SCRIPT_REPORT R
        RANGE SCRIPT_PROJECTS P
              ∀P  ∃R (R.SCRIPT_NO = P.SCRIPT_NO ∧
                      R.REPORT_SEQ = P.REPORT_SEQ)
              ∀P  ∃PR (PR.PROJ_NAME = P.PROJ_NAME)

RELATION SCRIPT_REPORT
        (FIELD SCRIPT_NO numeric(10)
        FIELD REPORT_SEQ numeric(3)
        FIELD REPORT_CODE char(10)
        FIELD REPORT_TYPE char(20)
        FIELD REPORT_TYPE_SELECTION char(10))
    KEY (SCRIPT_NO, REPORT_SEQ)
    CONSTRAINT
        RANGE PROJECT PROJ
        RANGE PERM_SCRIPT P
        RANGE TEMP_SCRIPT T
```

```
        RANGE SCRIPT_REPORT S
        RANGE VAL_REPORT_CODE VAL
              ∀S  P  T  (P.SCRIPT_NO = S.SCRIPT_NO
                         T.SCRIPT_NO = S.SCRIPT_NO)
              ∀S  VAL   (VAL.REPORT_CODE = S.REPORT_CODE)
              ∀S  PROJ  ((S.REPORT_TYPE SELECTION = 'INACTIVE'
                         V S.REPORT_TYPE SELECTION = 'ACT_MAINT'
                         V S.REPORT_TYPE SELECTION = 'ACT_DEV'
                         V S.REPORT_TYPE SELECTION = 'ALL'
                         V S.REPORT_TYPE SELECTION = 'LIST')
                         ∧ S.REPORT_TYPE = 'M') ∨
                         ((S.REPORT_TYPE_SELECTION = null)
                         ∧ (S.REPORT_TYPE = 'O')) V
                         (S.REPORT_TYPE SELECTION = PROJ.PROJ_NAME ∧
                         S.REPORT_TYPE = 'S')

RELATION SEQNO
        (FIELD TABLE_NAME char(30)
        FIELD FIELD_NAME char(30)
        FIELD MAXSEQNO numeric(10))
    KEY (TABLE_NAME, FIELD_NAME)
    CONSTRAINT
        RANGE SEQNO S
              ∀S  ∃S  (S.TABLE_NAME = a valid relation name
                       S.FIELD_NAME = a valid field name within that
                       relation)

RELATION SPECIAL ACT
        (FIELD EFF_ID numeric(10)
        (FIELD SP_ACTIVITY char(10)
        FIELD ACT_HR numeric(10, 2))
    KEY (EFF_ID, SP_ACTIVITY)
    CONSTRAINT
        RANGE SPECIAL_ACT SA
        RANGE EFF_PROJ EP
        RANGE EFF_SUB ES
        RANGE VAL_SP_ACTIVITY VAL
              ∀SA  ∃EP ES (EP.P_ID = SA.EFF_ID
                           ES.PS_ID = SA.EFF_ID)
              ∀SA  ∃VAL (VAL.SP_ACTIVITY = SA.SP_ACTIVITY)
```

```
RELATION SUBSYSTEM
        (FIELD SUBSY_ID numeric(5)
        FIELD NAME char(40)
        FIELD FUNCTION char(10))
    KEY (SUBSY_ID)
    CONSTRAINT
        RANGE PROJ_SUB S
        RANGE SUBSYSTEM SUB
        RANGE VAL_S FUNCTION VSF
                ∀SUB   ∃S (S.SUBSY_ID = SUB.SUBSY_ID)
                ∀SUB   ∃VSF (VSF.CODE = SUB.FUNCTION)

RELATION SUB_COM
        (FIELD SUBSY_ID numeric(5)
        FIELD COM_NAME char(40)
        FIELD COM_NO numeric(7)
        FIELD COM DATE date)
    KEY (SUBSY_ID, COM_NAME)
    UNIQUE (COM_NO)
    INDEX (COM_NO)
    CONSTRAINT
        RANGE PROJ_SUB S
        RANGE SUB_COM C
                ∀C   ∃S (S.SUBSY_ID = C.SUBSY_ID)

RELATION TABLE_PRIVILEGE
        (FIELD TABLE_NAME char(40)
        FIELD USER_CLASS char(20)
        FIELD SELECT_PRIV char(1)
        FIELD INSERT_PRIV char(1)
        FIELD UPDATE_PRIV char(1)
        FIELD DELETE_PRIV char(1)
        FIELD ALTER_PRIV char(1)
        FIELD INDEX_PRIV char(1))
    KEY (TABLE_NAME, USER_CLASS)
```

CONSTRAINT

RANGE TABLE_PRIVILEGE T

RANGE USER_CLASS U

$\forall T \quad \exists U$ (U.USER_CLASS = T.USER_CLASS)

$\forall T \quad \exists T\_$(T.TABLE_NAME = a valid relation in the database)

RELATION TEMP_ACTIVITY

(FIELD ACTIVITY char(10)

FIELD SAT_DAY date

FIELD HOURS numeric(10,2)

FIELD PROJ_NO numeric(3)

FIELD SUB_HR numeric(10,2)

FIELD FLAG char(4)

FIELD SCRIPT_NO numeric(10))

CONSTRAINT

RANGE TEMP_ACTIVITY TEMP

RANGE GENERATE_SAT_DAY GSAT

$\forall$TEMP $\exists$GSAT (GSAT.SCRIPT_NO = TEMP.SCRIPT_NO
$\wedge$ GSAT.SAT_DAY = TEMP.SAT_DAY)

RELATION TEMP_FORMCT

(FIELD SUB_DATE date

FIELD PROG_ID numeric(5)

FIELD FORM_TYPE char(6)

FIELD PROJ_NO numeric(3)

FIELD SCRIPT_NO numeric(10))

CONSTRAINT

RANGE TEMP_FORMCT TEMP

RANGE GENERATE_SAT_DAY GSAT

$\forall$TEMP $\exists$GSAT (GSAT.SCRIPT_NO = TEMP.SCRIPT_NO
$\wedge$ GSAT.SAT_DAY = TEMP.SAT_DAY)

RELATION TEMP_MANHRS

(FIELD FORM_NAME char(15)

FIELD SAT_DAY date

FIELD HOURS numeric(10,2)

FIELD PROJ_NO numeric(3)
FIELD PROG_ID numeric(5)
FIELD SUB_HR numeric(10,2)
FIELD FLAG char(4)
FIELD P_ID numeric(10)
FIELD SCRIPT_NO numeric(10))

CONSTRAINT
RANGE TEMP_MANHRS TEMP
RANGE GENERATE_SAT_DAY GSAT
∀TEMP ∃GSAT (GSAT.SCRIPT_NO = TEMP.SCRIPT_NO
∧ GSAT.SAT_DAY = TEMP.SAT_DAY)

RELATION TEMP_SCRIPT
(FIELD SCRIPT_NO numeric(10)
FIELD ORA_USER char(20)
FIELD PROCESS_ID char(20)
FIELD OUT_ROUTING char(20)
FIELD OUT_FILE char(20)
FIELD RUN_STATUS char(10)
FIELD DELETE_STATUS char(10))
KEY (SCRIPT_NO)
CONSTRAINT
RANGE USER_CLASS U
RANGE TEMP_SCRIPT T
∀T ∃U (U.ORA_USER = T.ORA_USER)
∀T ∃T ((T.OUT_ROUTING = '2' ∨ T.OUT_ROUTING = '1')
(T.OUT_FILE != null ∧ T.OUT_ROUTING = '1'))

RELATION TEMP_SERVHRS
(FIELD FORM_NAME char(15)
FIELD SAT_DAY date
FIELD HOURS numeric(10,2)
FIELD PROJ_NO numeric(3)
FIELD PROG_ID numeric(5)
FIELD FLAG char(4)
FIELD P_ID numeric(10)

```
        FIELD SCRIPT_NO numeric(10))
    CONSTRAINT
        RANGE TEMP_SERVHRS TEMP
        RANGE GENERATE_SAT_DAY GSAT
                ∀TEMP   ∃GSAT (GSAT.SCRIPT_NO = TEMP.SCRIPT_NO
                                ∧ GSAT.SAT_DAY = TEMP.SAT_DAY)

RELATION_USER_CLASS
        (FIELD ORA_USER_ID char(20)
        FIELD_USER_CLASS char(20))
    KEY (ORA_USER_ID)
    CONSTRAINT
        RANGE USER_CLASS_ACCESS UA
        RANGE USER_CLASS U
                ∀U   ∃U (U.ORA_USER_ID = a valid ORACLE user ID)
                ∀U   ∃UA (UA.USER_CLASS = U.USER_CLASS)

RELATION_USER_CLASS_ACCESS
        (FIELD_USER_CLASS char(20)
        FIELD ACCESS_TYPE char(10))
    KEY (USER_CLASS, ACCESS_TYPE)
    CONSTRAINT
        RANGE USER_CLASS_ACCESS UA
        RANGE USER_CLASS U
                ∀U      ∃UA (UA.USER_CLASS = U.USER_CLASS)
                ∀UA     ∃UA (UA.ACCESS_TYPE = ('BACKUP' ∨ 'DBA'
                                ∨ 'DELETE' ∨ 'DISTAPE' ∨ 'FORM' ∨ 'GENERAL'
                                ∨ 'IMPORT' ∨ 'INSERT' ∨ 'QA' ∨ 'QUERY'
                                ∨ 'REPORT' ∨ 'RESTORE' ∨ 'UPDATE' ∨ 'VIEW'))

RELATION VALIDATION
        (FIELD F_NAME char(20)
        FIELD CODE char(10)
        FIELD VALUE char(75))
    KEY (F_NAME, CODE)

VIEW AUTHORIZE
        (FIELD ACCESS_TYPE, SOURCE USER_CLASS_ACCESS
        FIELD ORA_USER_ID, SOURCE USER_CLASS)
```

VIEW VAL_ACTIVE_STATUS
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_ACTIVITY
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_ADA_FEATURE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_CH_CAUSE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_CH_CLASS
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_CH_OBJECT
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_CH_TYPE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_CL ACTIVITY
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_COM_CH
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_COM_PURPOSE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_COM_TYPE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_DATA_AVAIL
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_DSF_MEASURE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_DSF_STATUS
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_DSF_TARGET
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_ERR_ACAUSE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_ERR_ARES
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_ERR_CLASS
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_ERR_SOURCE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_ERR_TOOLS
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_FINAL_ORIGIN_CAT
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_ISO_CH
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_MAINT_ACT
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_MAINT_CH_TYPE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_MAINT_CLASS
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_MAINT_COM_CH
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_MAINT_ISO_CH
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_MEAS_TYPE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_NOTE_TYPE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_ORI_TYPE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_PHASE_CO
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_PROJ_TYPE
(FIELD PROJ_NO, SOURCE PROJECT
FIELD PROJ_TYPE, SOURCE PROJECT)

VIEW VAL_QA_STATUS
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_REPORT_CODE
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_SECOND_L
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_S_FUNCTION
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_SP_ACTIVITY
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW VAL_STATUS
(FIELD CODE, SOURCE VALIDATION
FIELD VALUE, SOURCE VALIDATION)

VIEW V_CLEANROOM_ACT
(FIELD EFF_ID, SOURCE EFF_ACT
FIELD_ACTIVITY, SOURCE EFF_ACT
FIELD ACT_HR, SOURCE EFF_ACT)
CONSTRAINT
RANGE EFF_ACT_EA

RANGE V_CLEANROOM_ACT VCA
RANGE VAL_CL_ACTIVITY VALA
∀VCA ∃EA ∃VALA (EA.ACTIVITY LIKE 'CL%' ∧
VALA.CODE = VCA.CL ACTIVITY)

VIEW V_CLEANROOM_PROJECTS
(FIELD PROJ_NAME, SOURCE PROJECT)

VIEW V_PERM_SCRIPT
(FIELD SCRIPT_NAME, SOURCE PERM_SCRIPT)

VIEW V_PROJ_COM
(FIELD PROJ_NAME, SOURCE PROJECT
FIELD SUB_PRE, SOURCE PROJ_SUB
FIELD COM_NAME, SOURCE SUB_COM
FIELD COM_NO, SOURCE SUB_COM)

VIEW V_PROJ_SUB_ACT
(FIELD PROJ_NAME, SOURCE PROJECT
FIELD SUB_PRE, SOURCE EFF_SUB
FIELD ACTIVITY, SOURCE EFF_ACT
FIELD ACT_HR, SOURCE EFF_ACT)

VIEW V_REP_CODES_CRITERIA
(FIELD VALUE, SOURCE REP CODES)

VIEW V_SEQNO
(FIELD TABLE_NAME, SOURCE SEQNO
FIELD FIELD_NAME, SOURCE SEQNO
FIELD MAXSEQNO, SOURCE SEQNO)

VIEW V_SUBSYSTEM_INFO
(FIELD FUNCTION, SOURCE SUBSYSTEM
FIELD NAME, SOURCE SUBSYSTEM
FIELD PROJ_NAME, SOURCE PROJECT
FIELD SUB_DATE, SOURCE PROJ_SUB
FIELD SUB_PRE, SOURCE PROJ_SUB)

GLOSSARY

Clause	A portion of a SQL command, starting with a reserved word, that qualifies or constrains the operation of the command.
Cluster	An internal mechanism for storing together groups of related columns from different tables, or groups of like-valued column entries from a single table, to improve efficiency. (There are no clusters in the SEL database.)
Column	A particular class of data items within a table. Each column has a single value in each row of a table. Also called a field.
Command	An instruction to the SQL*Plus interpreter.
Field	Synonymous with column.
Group	A SQL*Plus function that operates on a single column of all rows in a query, returning a single value.
Index	A mechanism for improving efficiency of database access by enabling searches to be performed without always examining an entire table.
Join	Retrieval of related rows from two or more tables in a single query.
Null	A "value" for a column indicating that the column has no value. Null values do not use storage space.
Order by	A SQL clause that controls the order of displayed rows.
Primary Key	A column or concatenation of columns whose values are frequently used to access a row of a table.
Query	An instruction to the SQL*Plus interpreter to retrieve one or more rows and columns from one or more tables or views.
Record	Synonymous with row.
Relation	Synonymous with table.
Row	A single entry in a table, containing one entry for each column in the table. Also called a record.
Subquery	A query enclosed in parentheses that returns values used in a condition of a SQL command.
Table	The basic unit of data storage in a relational DBMS. Contains a variable number of rows, each of which contains a fixed number of columns. Also called a relation.
View	A "virtual table" that consists of one or more columns from underlying database tables. Views do not actually store data.

ABBREVIATIONS AND ACRONYMS

AGSS	Attitude Ground Support System
CDR	critical design review
CLPRF	Cleanroom Personnel Resources Form
COF	Component Origination Form
CPU	central processing unit
CRF	Change Report Form
CSC	Computer Sciences Corporation
DAMSEL	Database Access Manager for the Software Engineering Laboratory
DBA	database administrator
DDL	data definition language
DSF	Development Status Form
ERRCO	error correction
FDF	Flight Dynamics Facility
GSFC	Goddard Space Flight Center
ID	identification
MCRF	Maintenance Change Report Form
NASA	National Aeronautics and Space Administration
OSMR	Operational Software Modification Number
PC	personal computer
PCSF	Project Completion Statistics Form
PDL	program design language
PDR	preliminary design review
PEF	Project Estimates Form
PMF	Project Message Form
PRF	Personnel Resources Form
PSF	Project Startup Form

QA	quality assurance
RDBMS	relational database management system
SEF	Subjective Evaluation Form
SEL	Software Engineering Laboratory
SFR	Software Failure Report
SIF	Subsystem Information Form
SLOC	source lines of code
SPF	Services/Products Form
SQL	structured query language
STL	Systems Technology Laboratory
WMEF	Weekly Maintenance Effort Form

REFERENCES

1. Software Engineering Laboratory, SEL-92-002, *Data Collection Procedures for the Software Engineering Laboratory (SEL) Database*, G. Heller, J. Valett, and M. Wild, March 1992

2. Computer Sciences Corporation, CSC/TM-87/6016, *Design of the Rehosted SEL Database*, M. So and G. Heller, March 1987

3. ———, CSC/SD-88/6019, *Database Access Manager for the Software Engineering Laboratory (DAMSEL) User's Guide*, M. Buhler, K. Pumphrey, and D. Spiegel, March 1990

4. ORACLE Corporation, *SQL*Plus User's Guide and Reference, Version 3*, L. Colston, 1989

5. ORACLE Corporation, *SQL Language Reference Manual, Version 6*, D. Cheu and B. Linden, 1990

6. C. J. Date, *An Introduction to Database Systems, 2nd ed.*, Addison Wesley, 1977

STANDARD BIBLIOGRAPHY OF SEL LITERATURE

The technical papers, memorandums, and documents listed in this bibliography are organized into two groups. The first group is composed of documents issued by the Software Engineering Laboratory (SEL) during its research and development activities. The second group includes materials that were published elsewhere but pertain to SEL activities.

SEL-ORIGINATED DOCUMENTS

SEL-76-001, *Proceedings From the First Summer Software Engineering Workshop*, August 1976

SEL-77-002, *Proceedings From the Second Summer Software Engineering Workshop*, September 1977

SEL-77-004, *A Demonstration of AXES for NAVPAK*, M. Hamilton and S. Zeldin, September 1977

SEL-77-005, *GSFC NAVPAK Design Specifications Languages Study*, P. A. Scheffer and C. E. Velez, October 1977

SEL-78-005, *Proceedings From the Third Summer Software Engineering Workshop*, September 1978

SEL-78-006, *GSFC Software Engineering Research Requirements Analysis Study*, P. A. Scheffer and C. E. Velez, November 1978

SEL-78-007, *Applicability of the Rayleigh Curve to the SEL Environment*, T. E. Mapp, December 1978

SEL-78-302, *FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 3)*, W. J. Decker, W. A. Taylor, et al., July 1986

SEL-79-002, *The Software Engineering Laboratory: Relationship Equations*, K. Freburger and V. R. Basili, May 1979

SEL-79-003, *Common Software Module Repository (CSMR) System Description and User's Guide*, C. E. Goorevich, A. L. Green, and S. R. Waligora, August 1979

SEL-79-004, *Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment*, C. E. Goorevich, A. L. Green, and W. J. Decker, September 1979

SEL-79-005, *Proceedings From the Fourth Summer Software Engineering Workshop*, November 1979

10000229
1016/0840

SEL-80-002, *Multi-Level Expression Design Language-Requirement Level (MEDL-R) System Evaluation*, W. J. Decker and C. E. Goorevich, May 1980

SEL-80-003, *Multimission Modular Spacecraft Ground Support Software System (MMS/GSSS) State-of-the-Art Computer Systems/Compatibility Study*, T. Welden, M. McClellan, and P. Liebertz, May 1980

SEL-80-005, *A Study of the Musa Reliability Model*, A. M. Miller, November 1980

SEL-80-006, *Proceedings From the Fifth Annual Software Engineering Workshop*, November 1980

SEL-80-007, *An Appraisal of Selected Cost/Resource Estimation Models for Software Systems*, J. F. Cook and F. E. McGarry, December 1980

SEL-80-008, *Tutorial on Models and Metrics for Software Management and Engineering*, V. R. Basili, 1980

SEL-81-008, *Cost and Reliability Estimation Models (CAREM) User's Guide*, J. F. Cook and E. Edwards, February 1981

SEL-81-009, *Software Engineering Laboratory Programmer Workbench Phase 1 Evaluation*, W. J. Decker and F. E. McGarry, March 1981

SEL-81-011, *Evaluating Software Development by Analysis of Change Data*, D. M. Weiss, November 1981

SEL-81-012, *The Rayleigh Curve as a Model for Effort Distribution Over the Life of Medium Scale Software Systems*, G. O. Picasso, December 1981

SEL-81-013, *Proceedings of the Sixth Annual Software Engineering Workshop*, December 1981

SEL-81-014, *Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL)*, A. L. Green, W. J. Decker, and F. E. McGarry, September 1981

SEL-81-101, *Guide to Data Collection*, V. E. Church, D. N. Card, F. E. McGarry, et al., August 1982

SEL-81-104, *The Software Engineering Laboratory*, D. N. Card, F. E. McGarry, G. Page, et al., February 1982

SEL-81-107, *Software Engineering Laboratory (SEL) Compendium of Tools (Revision 1)*, W. J. Decker, W. A. Taylor, E. J. Smith, et al., February 1982

SEL-81-110, *Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics*, G. Page, F. E. McGarry, and D. N. Card, June 1985

SEL-81-305, *Recommended Approach to Software Development*, L. Landis, F. E. McGarry, S. Waligora, et al., June 1992

SEL-82-001, *Evaluation of Management Measures of Software Development*, G. Page, D. N. Card, and F. E. McGarry, September 1982, vols. 1 and 2

SEL-82-004, *Collected Software Engineering Papers: Volume 1*, July 1982

SEL-82-007, *Proceedings of the Seventh Annual Software Engineering Workshop*, December 1982

SEL-82-008, *Evaluating Software Development by Analysis of Changes: The Data From the Software Engineering Laboratory*, V. R. Basili and D. M. Weiss, December 1982

SEL-82-102, *FORTRAN Static Source Code Analyzer Program (SAP) System Description (Revision 1)*, W. A. Taylor and W. J. Decker, April 1985

SEL-82-105, *Glossary of Software Engineering Laboratory Terms*, T. A. Babst, M. G. Rohleder, and F. E. McGarry, October 1983

SEL-82-1006, *Annotated Bibliography of Software Engineering Laboratory Literature*, L. Morusiewicz and J. Valett, November 1991

SEL-83-001, *An Approach to Software Cost Estimation*, F. E. McGarry, G. Page, D. N. Card, et al., February 1984

SEL-83-002, *Measures and Metrics for Software Development*, D. N. Card, F. E. McGarry, G. Page, et al., March 1984

SEL-83-003, *Collected Software Engineering Papers: Volume II*, November 1983

SEL-83-006, *Monitoring Software Development Through Dynamic Variables*, C. W. Doerflinger, November 1983

SEL-83-007, *Proceedings of the Eighth Annual Software Engineering Workshop*, November 1983

SEL-83-106, *Monitoring Software Development Through Dynamic Variables (Revision 1)*, C. W. Doerflinger, November 1989

SEL-84-003, *Investigation of Specification Measures for the Software Engineering Laboratory (SEL)*, W. W. Agresti, V. E. Church, and F. E. McGarry, December 1984

SEL-84-004, *Proceedings of the Ninth Annual Software Engineering Workshop*, November 1984

SEL-84-101, *Manager's Handbook for Software Development (Revision 1)*, L. Landis, F. E. McGarry, S. Waligora, et al., November 1990

SEL-85-001, *A Comparison of Software Verification Techniques*, D. N. Card, R. W. Selby, Jr., F. E. McGarry, et al., April 1985

SEL-85-002, *Ada Training Evaluation and Recommendations From the Gamma Ray Observatory Ada Development Team*, R. Murphy and M. Stark, October 1985

SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985

SEL-85-004, *Evaluations of Software Technologies: Testing, CLEANROOM, and Metrics*, R. W. Selby, Jr., and V. R. Basili, May 1985

SEL-85-005, *Software Verification and Testing*, D. N. Card, E. Edwards, F. McGarry, and C. Antle, December 1985

SEL-85-006, *Proceedings of the Tenth Annual Software Engineering Workshop*, December 1985

SEL-86-001, *Programmer's Handbook for Flight Dynamics Software Development*, R. Wood and E. Edwards, March 1986

SEL-86-002, *General Object-Oriented Software Development*, E. Seidewitz and M. Stark, August 1986

SEL-86-003, *Flight Dynamics System Software Development Environment (FDS/SDE) Tutorial*, J. Buell and P. Myers, July 1986

SEL-86-004, *Collected Software Engineering Papers: Volume IV*, November 1986

SEL-86-005, *Measuring Software Design*, D. N. Card et al., November 1986

SEL-86-006, *Proceedings of the Eleventh Annual Software Engineering Workshop*, December 1986

SEL-87-001, *Product Assurance Policies and Procedures for Flight Dynamics Software Development*, S. Perry et al., March 1987

SEL-87-002, *Ada® Style Guide (Version 1.1)*, E. Seidewitz et al., May 1987

SEL-87-003, *Guidelines for Applying the Composite Specification Model (CSM)*, W. W. Agresti, June 1987

SEL-87-004, *Assessing the Ada® Design Process and Its Implications: A Case Study*, S. Godfrey, C. Brophy, et al., July 1987

SEL-87-009, *Collected Software Engineering Papers: Volume V*, November 1987

SEL-87-010, *Proceedings of the Twelfth Annual Software Engineering Workshop*, December 1987

SEL-88-001, *System Testing of a Production Ada Project: The GRODY Study*, J. Seigle, L. Esker, and Y. Shi, November 1988

SEL-88-002, *Collected Software Engineering Papers: Volume VI*, November 1988

SEL-88-003, *Evolution of Ada Technology in the Flight Dynamics Area: Design Phase Analysis*, K. Quimby and L. Esker, December 1988

SEL-88-004, *Proceedings of the Thirteenth Annual Software Engineering Workshop*, November 1988

SEL-88-005, *Proceedings of the First NASA Ada User's Symposium*, December 1988

SEL-89-002, *Implementation of a Production Ada Project: The GRODY Study*, S. Godfrey and C. Brophy, September 1989

SEL-89-004, *Evolution of Ada Technology in the Flight Dynamics Area: Implementation/ Testing Phase Analysis*, K. Quimby, L. Esker, L. Smith, M. Stark, and F. McGarry, November 1989

SEL-89-005, *Lessons Learned in the Transition to Ada From FORTRAN at NASA/ Goddard*, C. Brophy, November 1989

SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989

SEL-89-007, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, November 1989

SEL-89-008, *Proceedings of the Second NASA Ada Users' Symposium*, November 1989

SEL-89-103, *Software Management Environment (SME) Concepts and Architecture (Revision 1)*, R. Hendrick, D. Kistler, and J. Valett, September 1992

SEL-89-201, *Software Engineering Laboratory (SEL) Database Organization and User's Guide (Revision 2)*, L. Morusiewicz and J. Bristow, October 1992

SEL-90-001, *Database Access Manager for the Software Engineering Laboratory (DAMSEL) User's Guide*, M. Buhler, K. Pumphrey, and D. Spiegel, March 1990

SEL-90-002, *The Cleanroom Case Study in the Software Engineering Laboratory: Project Description and Early Analysis*, S. Green et al., March 1990

SEL-90-003, *A Study of the Portability of an Ada System in the Software Engineering Laboratory (SEL)*, L. O. Jun and S. R. Valett, June 1990

SEL-90-004, *Gamma Ray Observatory Dynamics Simulator in Ada (GRODY) Experiment Summary*, T. McDermott and M. Stark, September 1990

SEL-90-005, *Collected Software Engineering Papers: Volume VIII*, November 1990

SEL-90-006, *Proceedings of the Fifteenth Annual Software Engineering Workshop*, November 1990

SEL-91-001, *Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules*, W. Decker, R. Hendrick, and J. Valett, February 1991

10000229
1016/0840

SEL-91-003, *Software Engineering Laboratory (SEL) Ada Performance Study Report*, E. W. Booth and M. E. Stark, July 1991

SEL-91-004, *Software Engineering Laboratory (SEL) Cleanroom Process Model*, S. Green, November 1991

SEL-91-005, *Collected Software Engineering Papers: Volume IX*, November 1991

SEL-91-006, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, December 1991

SEL-91-102, *Software Engineering Laboratory (SEL) Data and Information Policy (Revision 1)*, F. McGarry, August 1991

SEL-92-001, *Software Management Environment (SME) Installation Guide*, D. Kistler, January 1992

SEL-92-002, *Data Collection Procedures for the Software Engineering Laboratory (SEL) Database*, G. Heller, March 1992

SEL-RELATED LITERATURE

[4]Agresti, W. W., V. E. Church, D. N. Card, and P. L. Lo, "Designing With Ada for Satellite Simulation: A Case Study," *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986

[2]Agresti, W. W., F. E. McGarry, D. N. Card, et al., "Measuring Software Technology," *Program Transformation and Programming Environments*. New York: Springer-Verlag, 1984

[1]Bailey, J. W., and V. R. Basili, "A Meta-Model for Software Development Resource Expenditures," *Proceedings of the Fifth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1981

[8]Bailey, J. W., and V. R. Basili, "Software Reclamation: Improving Post-Development Reusability," *Proceedings of the Eighth Annual National Conference on Ada Technology*, March 1990

[1]Basili, V. R., "Models and Metrics for Software Management and Engineering," *ASME Advances in Computer Technology*, January 1980, vol. 1

Basili, V. R., *Tutorial on Models and Metrics for Software Management and Engineering*. New York: IEEE Computer Society Press, 1980 (also designated SEL-80-008)

[3]Basili, V. R., "Quantitative Evaluation of Software Methodology," *Proceedings of the First Pan-Pacific Computer Conference*, September 1985

[7]Basili, V. R., *Maintenance = Reuse-Oriented Software Development*, University of Maryland, Technical Report TR-2244, May 1989

10000229
1016/0840

[7]Basili, V. R., *Software Development: A Paradigm for the Future*, University of Maryland, Technical Report TR-2263, June 1989

[8]Basili, V. R., "Viewing Maintenance of Reuse-Oriented Software Development," *IEEE Software*, January 1990

[1]Basili, V. R., and J. Beane, "Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problems?," *Journal of Systems and Software*, February 1981, vol. 2, no. 1

[9]Basili, V. R., and G. Caldiera, *A Reference Architecture for the Component Factory*, University of Maryland, Technical Report TR-2607, March 1991

[1]Basili, V. R., and K. Freburger, "Programming Measurement and Estimation in the Software Engineering Laboratory," *Journal of Systems and Software*, February 1981, vol. 2, no. 1

[3]Basili, V. R., and N. M. Panlilio-Yap, "Finding Relationships Between Effort and Other Variables in the SEL," *Proceedings of the International Computer Software and Applications Conference*, October 1985

[4]Basili, V. R., and D. Patnaik, *A Study on Fault Prediction and Reliability Assessment in the SEL Environment*, University of Maryland, Technical Report TR-1699, August 1986

[2]Basili, V. R., and B. T. Perricone, "Software Errors and Complexity: An Empirical Investigation," *Communications of the ACM*, January 1984, vol. 27, no. 1

[1]Basili, V. R., and T. Phillips, "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," *Proceedings of the ACM SIGMETRICS Symposium/ Workshop: Quality Metrics*, March 1981

[3]Basili, V. R., and C. L. Ramsey, "ARROWSMITH-P—A Prototype Expert System for Software Engineering Management," *Proceedings of the IEEE/MITRE Expert Systems in Government Symposium*, October 1985

Basili, V. R., and J. Ramsey, *Structural Coverage of Functional Testing*, University of Maryland, Technical Report TR-1442, September 1984

Basili, V. R., and R. Reiter, "Evaluating Automatable Measures for Software Development," *Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity, and Cost*. New York: IEEE Computer Society Press, 1979

[5]Basili, V. R., and H. D. Rombach, "Tailoring the Software Process to Project Goals and Environments," *Proceedings of the 9th International Conference on Software Engineering*, March 1987

[5]Basili, V. R., and H. D. Rombach, "T A M E: Tailoring an Ada Measurement Environment," *Proceedings of the Joint Ada Conference*, March 1987

10000229
1016/0840

[5]Basili, V. R., and H. D. Rombach, "T A M E: Integrating Measurement Into Software Environments," University of Maryland, Technical Report TR-1764, June 1987

[6]Basili, V. R., and H. D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments," *IEEE Transactions on Software Engineering*, June 1988

[7]Basili, V. R., and H. D. Rombach, *Towards A Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment*, University of Maryland, Technical Report TR-2158, December 1988

[8]Basili, V. R., and H. D. Rombach, *Towards A Comprehensive Framework for Reuse: Model-Based Reuse Characterization Schemes*, University of Maryland, Technical Report TR-2446, April 1990

[9]Basili, V. R., and H. D. Rombach, *Support for Comprehensive Reuse*, University of Maryland, Technical Report TR-2606, February 1991

[3]Basili, V. R., and R. W. Selby, Jr., "Calculation and Use of an Environment's Characteristic Software Metric Set," *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985

Basili, V. R., and R. W. Selby, Jr., *Comparing the Effectiveness of Software Testing Strategies*, University of Maryland, Technical Report TR-1501, May 1985

[3]Basili, V. R., and R. W. Selby, Jr., "Four Applications of a Software Data Collection and Analysis Methodology," *Proceedings of the NATO Advanced Study Institute*, August 1985

[5]Basili, V. R., and R. Selby, "Comparing the Effectiveness of Software Testing Strategies," *IEEE Transactions on Software Engineering*, December 1987

[9]Basili, V. R., and R. W. Selby, "Paradigms for Experimentation and Empirical Studies in Software Engineering," *Reliability Engineering and System Safety*, January 1991

[4]Basili, V. R., R. W. Selby, Jr., and D. H. Hutchens, "Experimentation in Software Engineering," *IEEE Transactions on Software Engineering*, July 1986

[2]Basili, V. R., R. W. Selby, and T. Phillips, "Metric Analysis and Data Validation Across FORTRAN Projects," *IEEE Transactions on Software Engineering*, November 1983

[2]Basili, V. R., and D. M. Weiss, *A Methodology for Collecting Valid Software Engineering Data*, University of Maryland, Technical Report TR-1235, December 1982

[3]Basili, V. R., and D. M. Weiss, "A Methodology for Collecting Valid Software Engineering Data," *IEEE Transactions on Software Engineering*, November 1984

[1]Basili, V. R., and M. V. Zelkowitz, "The Software Engineering Laboratory: Objectives," *Proceedings of the Fifteenth Annual Conference on Computer Personnel Research*, August 1977

Basili, V. R., and M. V. Zelkowitz, "Designing a Software Measurement Experiment," *Proceedings of the Software Life Cycle Management Workshop*, September 1977

[1]Basili, V. R., and M. V. Zelkowitz, "Operation of the Software Engineering Laboratory," *Proceedings of the Second Software Life Cycle Management Workshop*, August 1978

[1]Basili, V. R., and M. V. Zelkowitz, "Measuring Software Development Characteristics in the Local Environment," *Computers and Structures*, August 1978, vol. 10

Basili, V. R., and M. V. Zelkowitz, "Analyzing Medium Scale Software Development," *Proceedings of the Third International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1978

[9]Booth, E. W., and M. E. Stark, "Designing Configurable Software: COMPASS Implementation Concepts," *Proceedings of Tri-Ada 1991*, October 1991

[9]Briand, L. C., V. R. Basili, and W. M. Thomas, *A Pattern Recognition Approach for Software Engineering Data Analysis*, University of Maryland, Technical Report TR-2672, May 1991

[5]Brophy, C. E., W. W. Agresti, and V. R. Basili, "Lessons Learned in Use of Ada-Oriented Design Methods," *Proceedings of the Joint Ada Conference*, March 1987

[6]Brophy, C. E., S. Godfrey, W. W. Agresti, and V. R. Basili, "Lessons Learned in the Implementation Phase of a Large Ada Project," *Proceedings of the Washington Ada Technical Conference*, March 1988

[2]Card, D. N., "Early Estimation of Resource Expenditures and Program Size," Computer Sciences Corporation, Technical Memorandum, June 1982

[2]Card, D. N., "Comparison of Regression Modeling Techniques for Resource Estimation," Computer Sciences Corporation, Technical Memorandum, November 1982

[3]Card, D. N., "A Software Technology Evaluation Program," *Annais do XVIII Congresso Nacional de Informatica*, October 1985

[5]Card, D. N., and W. W. Agresti, "Resolving the Software Science Anomaly," *The Journal of Systems and Software*, 1987

[6]Card, D. N., and W. W. Agresti, "Measuring Software Design Complexity," *The Journal of Systems and Software*, June 1988

[4]Card, D. N., V. E. Church, and W. W. Agresti, "An Empirical Study of Software Design Practices," *IEEE Transactions on Software Engineering*, February 1986

Card, D. N., V. E. Church, W. W. Agresti, and Q. L. Jordan, "A Software Engineering View of Flight Dynamics Analysis System," Parts I and II, Computer Sciences Corporation, Technical Memorandum, February 1984

Card, D. N., Q. L. Jordan, and V. E. Church, "Characteristics of FORTRAN Modules," Computer Sciences Corporation, Technical Memorandum, June 1984

[5]Card, D. N., F. E. McGarry, and G. T. Page, "Evaluating Software Engineering Technologies," *IEEE Transactions on Software Engineering*, July 1987

[3]Card, D. N., G. T. Page, and F. E. McGarry, "Criteria for Software Modularization," *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985

[1]Chen, E., and M. V. Zelkowitz, "Use of Cluster Analysis To Evaluate Software Engineering Methodologies," *Proceedings of the Fifth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1981

[4]Church, V. E., D. N. Card, W. W. Agresti, and Q. L. Jordan, "An Approach for Assessing Software Prototypes," *ACM Software Engineering Notes*, July 1986

[2]Doerflinger, C. W., and V. R. Basili, "Monitoring Software Development Through Dynamic Variables," *Proceedings of the Seventh International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1983

Doubleday, D., *ASAP: An Ada Static Source Code Analyzer Program*, University of Maryland, Technical Report TR-1895, August 1987 (NOTE: 100 pages long)

[6]Godfrey, S., and C. Brophy, "Experiences in the Implementation of a Large Ada Project," *Proceedings of the 1988 Washington Ada Symposium*, June 1988

Hamilton, M., and S. Zeldin, *A Demonstration of AXES for NAVPAK*, Higher Order Software, Inc., TR-9, September 1977 (also designated SEL-77-005)

[5]Jeffery, D. R., and V. Basili, *Characterizing Resource Data: A Model for Logical Association of Software Data*, University of Maryland, Technical Report TR-1848, May 1987

[6]Jeffery, D. R., and V. R. Basili, "Validating the TAME Resource Data Model," *Proceedings of the Tenth International Conference on Software Engineering*, April 1988

[5]Mark, L., and H. D. Rombach, *A Meta Information Base for Software Engineering*, University of Maryland, Technical Report TR-1765, July 1987

[6]Mark, L., and H. D. Rombach, "Generating Customized Software Engineering Information Bases From Software Process and Product Specifications," *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, January 1989

[5]McGarry, F. E., and W. W. Agresti, "Measuring Ada for Software Development in the Software Engineering Laboratory (SEL)," *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, January 1988

[7]McGarry, F., L. Esker, and K. Quimby, "Evolution of Ada Technology in a Production Software Environment," *Proceedings of the Sixth Washington Ada Symposium (WADAS)*, June 1989

[3]McGarry, F. E., J. Valett, and D. Hall, "Measuring the Impact of Computer Resource Quality on the Software Development Process and Product," *Proceedings of the Hawaiian International Conference on System Sciences*, January 1985

National Aeronautics and Space Administration (NASA), *NASA Software Research Technology Workshop* (Proceedings), March 1980

[3]Page, G., F. E. McGarry, and D. N. Card, "A Practical Experience With Independent Verification and Validation," *Proceedings of the Eighth International Computer Software and Applications Conference*, November 1984

[5]Ramsey, C. L., and V. R. Basili, *An Evaluation of Expert Systems for Software Engineering Management*, University of Maryland, Technical Report TR-1708, September 1986

[3]Ramsey, J., and V. R. Basili, "Analyzing the Test Process Using Structural Coverage," *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985

[5]Rombach, H. D., "A Controlled Experiment on the Impact of Software Structure on Maintainability," *IEEE Transactions on Software Engineering*, March 1987

[8]Rombach, H. D., "Design Measurement: Some Lessons Learned," *IEEE Software*, March 1990

[9]Rombach, H. D., "Software Reuse: A Key to the Maintenance Problem," *Butterworth Journal of Information and Software Technology*, January/February 1991

[6]Rombach, H. D., and V. R. Basili, "Quantitative Assessment of Maintenance: An Industrial Case Study," *Proceedings From the Conference on Software Maintenance*, September 1987

[6]Rombach, H. D., and L. Mark, "Software Process and Product Specifications: A Basis for Generating Customized SE Information Bases," *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, January 1989

[7]Rombach, H. D., and B. T. Ulery, *Establishing a Measurement Based Maintenance Improvement Program: Lessons Learned in the SEL*, University of Maryland, Technical Report TR-2252, May 1989

[6]Seidewitz, E., "Object-Oriented Programming in Smalltalk and Ada," *Proceedings of the 1987 Conference on Object-Oriented Programming Systems, Languages, and Applications*, October 1987

[5]Seidewitz, E., "General Object-Oriented Software Development: Background and Experience," *Proceedings of the 21st Hawaii International Conference on System Sciences*, January 1988

[6]Seidewitz, E., "General Object-Oriented Software Development with Ada: A Life Cycle Approach," *Proceedings of the CASE Technology Conference*, April 1988

[9]Seidewitz, E., "Object-Oriented Programming Through Type Extension in Ada 9X," *Ada Letters*, March/April 1991

[4]Seidewitz, E., and M. Stark, "Towards a General Object-Oriented Software Development Methodology," *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986

[9]Seidewitz, E., and M. Stark, "An Object-Oriented Approach to Parameterized Software in Ada," *Proceedings of the Eighth Washington Ada Symposium*, June 1991

[8]Stark, M., "On Designing Parametrized Systems Using Ada," *Proceedings of the Seventh Washington Ada Symposium*, June 1990

[7]Stark, M. E. and E. W. Booth, "Using Ada to Maximize Verbatim Software Reuse," *Proceedings of TRI-Ada 1989*, October 1989

[5]Stark, M., and E. Seidewitz, "Towards a General Object-Oriented Ada Lifecycle," *Proceedings of the Joint Ada Conference*, March 1987

[8]Straub, P. A., and M. V. Zelkowitz, "PUC: A Functional Specification Language for Ada," *Proceedings of the Tenth International Conference of the Chilean Computer Science Society*, July 1990

[7]Sunazuka, T., and V. R. Basili, *Integrating Automated Support for a Software Management Cycle Into the TAME System*, University of Maryland, Technical Report TR-2289, July 1989

Turner, C., and G. Caron, *A Comparison of RADC and NASA/SEL Software Development Data*, Data and Analysis Center for Software, Special Publication, May 1981

Turner, C., G. Caron, and G. Brement, *NASA/SEL Data Compendium*, Data and Analysis Center for Software, Special Publication, April 1981

[5]Valett, J. D., and F. E. McGarry, "A Summary of Software Measurement Experiences in the Software Engineering Laboratory," *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, January 1988

[3]Weiss, D. M., and V. R. Basili, "Evaluating Software Development by Analysis of Changes: Some Data From the Software Engineering Laboratory," *IEEE Transactions on Software Engineering*, February 1985

[5]Wu, L., V. R. Basili, and K. Reed, "A Structure Coverage Tool for Ada Software Systems," *Proceedings of the Joint Ada Conference*, March 1987

[1]Zelkowitz, M. V., "Resource Estimation for Medium-Scale Software Projects," *Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science*. New York: IEEE Computer Society Press, 1979

[2]Zelkowitz, M. V., "Data Collection and Evaluation for Experimental Computer Science Research," *Empirical Foundations for Computer and Information Science* (Proceedings), November 1982

[6]Zelkowitz, M. V., "The Effectiveness of Software Prototyping: A Case Study," *Proceedings of the 26th Annual Technical Symposium of the Washington, D. C., Chapter of the ACM*, June 1987

[6]Zelkowitz, M. V., "Resource Utilization During Software Development," *Journal of Systems and Software*, 1988

[8]Zelkowitz, M. V., "Evolution Towards Specifications Environment: Experiences With Syntax Editors," *Information and Software Technology*, April 1990

Zelkowitz, M. V., and V. R. Basili, "Operational Aspects of a Software Measurement Facility," *Proceedings of the Software Life Cycle Management Workshop*, September 1977

NOTES:

[1]This article also appears in SEL-82-004, *Collected Software Engineering Papers: Volume I*, July 1982.

[2]This article also appears in SEL-83-003, *Collected Software Engineering Papers: Volume II*, November 1983.

[3]This article also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.

[4]This article also appears in SEL-86-004, *Collected Software Engineering Papers: Volume IV*, November 1986.

[5]This article also appears in SEL-87-009, *Collected Software Engineering Papers: Volume V*, November 1987.

[6]This article also appears in SEL-88-002, *Collected Software Engineering Papers: Volume VI*, November 1988.

[7]This article also appears in SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989.

[8]This article also appears in SEL-90-005, *Collected Software Engineering Papers: Volume VIII*, November 1990.

[9]This article also appears in SEL-91-005, *Collected Software Engineering Papers: Volume IX*, November 1991.