SOFTWARE ENGINEERING LABORATORY SERIES

SEL-89-001

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

MAY 1989

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 for the purpose of investigating the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1977 and has three primary organizational members:

NASA/GSFC, Systems Development Branch
The University of Maryland, Computer Sciences Department
Computer Sciences Corporation, Systems Development 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 major contributors to this document are

Maria So	(Computer Sciences Corporation)
Gerard Heller	(Computer Sciences Corporation)
Sandra Steinberg	(Computer Sciences Corporation)
Douglas Spiegel	(Goddard Space Flight Center)

Single copies of this document can be obtained by writing to

Systems Development Branch
Code 552
Goddard Space Flight Center
Greenbelt, Maryland 20771

ABSTRACT

The organization of the Software Engineering Laboratory (SEL) database is presented. 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

TABLE OF CONTENTS (Cont'd)

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

SECTION 1 - INTRODUCTION

The Software Engineering Laboratory (SEL) was established in 1977 to support research in the 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 and clusters 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 DBMS. 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 is a glossary of database-specific terms and abbreviations; Appendix D presents the SEL data collection forms; and Appendix E contains the data definition language (DDL) that specifies the definitions of tables, views, and all the constraints needed to maintain data integrity in the SEL database environment.

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. Each row consists of a fixed number of columns, or fields. Columns are identified by column names and may contain values of a particular data type (e.g., character, number, date). The columns contain both the actual data being stored and data that 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. There is no predefined order in which the rows of a table are stored. In most tables, a particular column or group of columns is defined as the primary key of the table. This means that the values of those columns will be unique for every row in the table. There may also be columns other than the primary key that must be unique across all rows. 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 might be the project name column (PROJ_NAME), while the primary key in the PROJ_SUB table might be 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

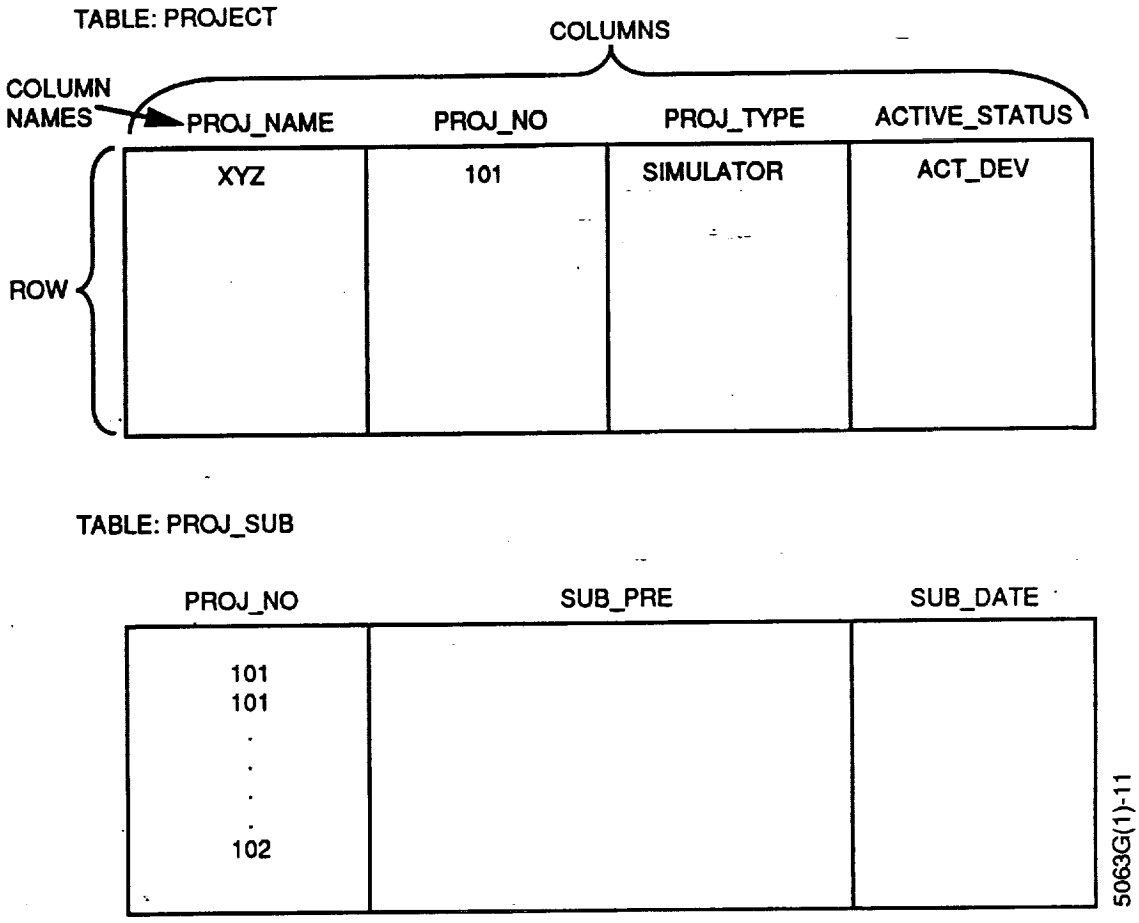


Figure 1-1. Basic Relational Database Organization

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).

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 names, services name, 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 to refer to them 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.

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 are also collected during the maintenance stage. The following sections describe data types that characterize the development 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

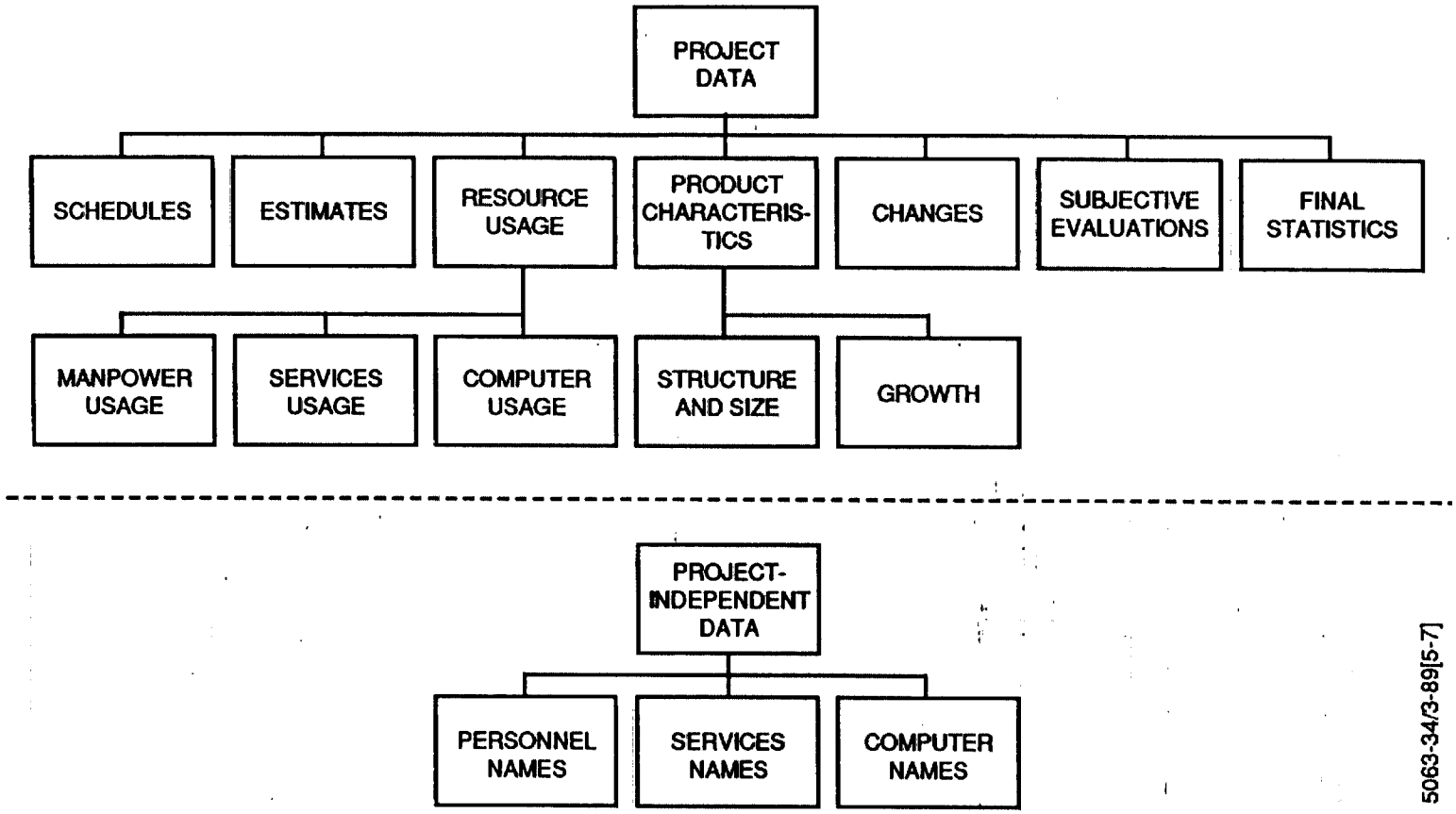


Figure 2-1. Conceptual View of SEL Data

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

- General notes on project or data peculiarities
- Contacts for the project
- Name of the project controlled source library
- SEL forms collected for the project
- Computer on which project is being developed
- Project task numbers
- Tools used for collecting project data

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. When a project completes the development stage, a final schedule is submitted that reflects the actual schedule that was followed by the project. Schedule data exist in sets that include the following:

P1 - Project name

P5 - Submission date of the current schedule

P6 - Requirements definition phase start and end dates

P7 - Design phase start and end dates

P8 - Code and test (implementation) 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

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 estimates 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 - Submission date of the current set of estimates
- P14 - Number of subsystems in the software product
- P15 - Number of components in the software product
- P16 - Total lines of code in the software product
- P17 - Old lines of code in the software product
- P18 - Modified lines of code in the software product
- P19 - New lines of code 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 are individual routines or modules that are maintained in separate files. (See Reference 1 for a more detailed description of these concepts.)

The lines-of-code estimates collected 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. Programmer hours is the estimate of the total technical effort spent on the project. Similarly, management hours is the estimate of the total hours spent

directly managing the project. Services hours refers to the estimated hours spent by support personnel on the project. This includes secretaries, technical editors, word processors, data librarians, couriers, and indirect levels of project management.

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.

2.1.3.1 Manpower

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.

In addition, for projects that began before June 1987, the manpower resource hours may be further classified by the subsystem on which the work was 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 - Programmer name; name of the person performing technical or 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/review design hours; hours spent reading and reviewing design materials (peer reviews, design walk-throughs, etc.)

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

P29 - Read/review code hours; hours spent reading code for any purpose except isolation of errors (peer review, code walk-throughs, 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 as being associated with a particular subsystem of the project. 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 weekly activity hours, programmer hours are recorded categorized by 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/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 represents the total hours worked by a given programmer during a given week.

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

2.1.3.2 Services

Each week during the development stage of a project, service 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 service hours recorded each week for a given project are as follows:

P1 - Project name

P23 - Week ending date

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

P40 - Secretary hours; hours spent by secretaries in support of technical and management-related project paperwork

P41 - Librarians; hours spent by data librarians in support of the project (includes data entry, tape generation, etc.)

P42 - Program 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

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 log-ons 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 - Subsystem date; date on which the subsystem information was entered into the database

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:

P1 - Project name

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

P51 - Component name; mnemonic name used in identifying the component

P52 - Component date; date on which the component information was entered into the database

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

P54 - Submission date; date on which the component information was recorded by the programmer

P55 - Programmer name; name of programmer who created the component

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). 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

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

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

P62 - Changes; count of the number of changes that have occurred in the project 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

Detailed information is recorded in the database for each change that takes place in a project's configured software. 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

P64 - Programmer name; name of the programmer implementing the change

P65 - Submission 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; list of the names of the components that were changed

P85 - Ada features; list of the Ada features that were involved in an error (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)

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, measurements are recorded of the actual values of parameters that were estimated earlier and of additional parameters that were not estimated. 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 - Submission date of final statistics

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 the system

P140 - Number of pages of documentation produced for the system

P141 - Total source lines of code in the system

P142 - Total newly created lines of code in the system

P143 - Total lines of code in the system that were modifications to existing code from other systems

P144 - Total lines of code in the system that were used from other systems without modification

P145 - Total number of comment lines in the source code

P146 - Total number of executable modules in the system

P147 - Total newly created executable modules in the system

P148 - Total executable modules in the system that were modified from other systems

P149 - Total executable modules in the system that were used from other systems without modification

P150 - Total number of executable lines of code in the system

P151 - Total newly created executable lines of code in the system

P152 - Total executable lines of code in the system that were modified from other systems

P153 - Total executable lines of code in the system that were used from other systems without modification

and for each executable component in the system:

P154 - Number of executable statements in the component

P155 - Total number of source lines in the component

P156 - Total number of comment lines in the component

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 resource hours are recorded

or who submits component or change information is represented in the database by the following data items:

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

M2 - Full name; programmer's complete first and last name

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

Service personnel are stored in the database as generic programmers; 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 COMPUTERS

The other class of support entity 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 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 D. 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 Project Estimates Form (PEF) (Figure D-1 in Appendix D) 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 (P13)

D3 - Requirements; estimated requirements definition phase start date

D4 - Design; estimated design phase start date

D5 - Code and test; estimated code and test (implementation) phase start date

D6 - System test; estimated system test phase start date

D7 - Acceptance test; estimated acceptance test phase start date

D8 - Cleanup; estimated cleanup phase start date

D9 - Maintenance; estimated maintenance stage start date

D10 - Project end; estimated project end date

D11 - Programmer hours (P20)

D12 - Management hours (P21)

D13 - Service hours (P22)

D14 - Number of subsystems (P14)

D15 - Number of components (P15)

D16 - Total lines (P16)

D17 - New lines (P19)

D18 - Modified lines (P18)

D19 - Old lines (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) and the Services/Products Form (SPF) provide weekly rate information for the projects. The PRF, Figure D-2, captures the actual technical/management expenditure history on the project. This form also contains information on the type of activity on which the manpower hours were spent during the week. A separate section of the form is used to record hours spent performing specific activities that are of current interest to the SEL.

The PRF 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 - Programmer 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 - Debug 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 SPF, Figure D-3, measures resource expenditure in support personnel hours 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 IBM 4341 and the VAX 11/780. The computer information is taken from computer accounting reports from these computers. Service 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.

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 modules (P61)

D42 - Number of changes (P62)

D43 - Lines of code (P60)

D44 - Technical publications hours (P39)

D45 - Secretary hours (P40)

D46 - Librarians' hours (P41)

D47 - Other hours (P43)

D48 - Project management hours (P42)

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

3.1.3 PRODUCT DATA FORMS

The Component Origination Form (COF), the Change Report Form (CRF), and the Subsystem Information Form (SIF) provide product data information for the project. The COF, Figure D-4, records information about the components in the system. Some of the information collected is 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 controlled library. A form is completed for each component in the system at the time when the component is ready to be moved into the project controlled source library.

COF FIELDS

D1 - Project name (P1)

D50 - Programmer name (P55)

D51 - Subsystem prefix (P47)

D52 - Form date (P54)

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 D-5, contains information about the type of change that was made, the components that were changed, error information if applicable, and Ada-specific information if applicable. The CRF is completed by personnel who implement changes to the system that involve modifying components in the project-controlled source library. A form is submitted for each change to the system at the time the changed components are updated in the project-controlled source library.

CRF FIELDS

D1 - Project name (P1)

D60 - Current date (P65)

D61 - Programmer name (P64)

D62 - 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 used (P85)

D78 - Documentation understandable (P82)

D79 - Which statements are true (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 SIF, Figure D-6, 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)
D151 - Subsystem date (P50)
D152 - Subsystem prefix (P47)
D153 - Subsystem name (P48)
D154 - Subsystem function (P49)

3.1.4 PROJECT COMPLETION FORMS

The Project Completion Statistics Form (PCSF) and the Subjective Evaluation Form (SEF) provide project completion information for completed projects. The PCSF, Figure D-7, is used to record the final statistics for the project. This information includes the actual project resources expenditures, project schedule, and the software product size.

The PCSF is completed by the project leader. It is submitted when the final system products have been delivered. 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)

D83 - Form date (P124)

D84 - Requirements; actual requirements definition phase start date

D85 - Design; actual design phase start date

D86 - Code and test; actual code and test (implementation) phase start date

D87 - System test; actual system test phase start date

D88 - Acceptance test; actual acceptance test phase start date

D89 - Cleanup; actual cleanup phase start date

D90 - Maintenance; actual maintenance stage start date

D91 - Project end; actual project end date

D92 - Technical and management hours (P132)

D93 - Service 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 source lines of code (P141)

D101 - New source lines of code (P142)

D102 - Modified source lines of code (P143)

D103 - Old source lines of code (P144)

D104 - Comments (P145)

D105 - Total executable modules (P146)

D106 - New executable modules (P147)

D107 - Modified executable modules (P148)

D108 - Old executable modules (P149)

D109 - Total executable statements (P150)

D110 - New executable statements (P151)

D111 - Modified executable statements (P152)

D112 - Old executable statements (P153)

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

The SEF, Figure D-8, 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 - Submission date (P13)

D21 - Project personnel name (P24)

D114 - Problem difficulty/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 change/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 (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/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

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.

4.1 DATABASE TABLE AND VIEW DEFINITIONS

The SEL database contains a total of 48 base tables (relations) and 30 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 present the tables and views in the database and their component fields. Table 4-1, which contains 32 tables and 3 views, is intended for all database users. The additional tables and views that are not included in this table are mainly used for data entry and system maintenance. Table 4-1 presents, for each table and view, the table or view name; the name of each column; a description of each table and column; the type of each column and its length; a list of valid values for columns where coded values are used; and one or more reference IDs for most columns, that cross-reference the column to data item descriptions in Sections 2 and 3. A translation of the codes used in Table 4-1 can be found in Appendix A. Columns that are part of the primary key are underlined, columns that do not have reference IDs are generally internal identifiers

used for relating tables to one another. The data types for columns may be one of the following: char, number, and date. A char column that may contain a sequence of alphanumeric is followed by the maximum length of the field. A number column that may contain numerals is followed by the width of the field and the number of decimal places, if applicable. A date column may contain a date formatted as DD-MMM-YY. Reference 4 presents a more detailed description of various data types.

Table 4-2 is intended for users, such as maintenance programmers, who need to know more of the technical specifications for all 43 base tables and 27 views. Provided for each field are its name; its data type; its length and the number of decimal places if it is a numeric field; an indication of whether it is part of the primary key; and a specification of whether it can contain null values, whether it is indexed, and whether it is clustered with another table. 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. 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 are created for all the fields that must have unique values within a particular table. All the primary keys are also uniquely indexed.

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 defining that data and describing its 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 relational database environment, tables are stored without predefined orders. Due to the semantics of the data itself, however, tables do 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

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

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, D62
	PROG_ID	ID UNIQUELY IDENTIFYING EACH PROGRAMMER	NUMBER (5, 0)		
	SUB_DATE	SUBMISSION DATE OF CRF	DATE		P65, D60
	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 OR NOT	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 (2, 0)		P74
	NUM_COM_EX	TOTAL NUMBER OF COMPONENTS EXAMINED	NUMBER (2, 0)		P75
	CH_TYPE	TYPE OF CHANGE	CHAR (10)	ERRCO, PLANE, IMPRE, IMPCM, IMPUS, INDE, 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	
CHANGE_COM		TABLE CONTAINING CHANGED COMPONENTS ASSOCIATED WITH PARTICULAR CRFs			
	CHANGE_NO	FORM NUMBER OF CRF	CHAR (6)		P63, D62
	COM_NO	ID OF CHANGED COMPONENT	NUMBER (7, 0)		
CH_ADAFEAT		TABLE CONTAINING ADA FEATURES THAT WERE INVOLVED IN OR CONTRIBUTED TO PARTICULAR CHANGES			
	CHANGE_NO	FORM NUMBER OF CRF	CHAR (6)		P63, D62
	ADA_FEATURE	FEATURE(S) INVOLVED IN CHANGE IF ADA IS USED AS DESIGN AND IMPLEMENTATION LANGUAGE	CHAR (10)	DATATYPE, SUBPROG, EXCEPT, GEN, PACK, TASK, SYSDEPF, OTHER	P65, D77

5063G-(6)-24

Table 4-1. SEL Database Tables and Views--Table and Column Descriptions (2 of 9)

TABLE OR VIEW NAME	COLUMN NAME	DESCRIPTION	TYPE	VALID CODE/VALUE	REFERENCE ID
CH_ERR_ARES		TABLE CONTAINING RESOURCES USED IN CORRECTING ERRORS FOR PARTICULAR CHANGES INVOLVING ADA			
	CHANGE_NO	FORM NUMBER OF CRF	CHAR (6)		P63, D62
	ERR_ARES	RESOURCES USED TO CORRECT ERROR CAUSED BY USE OF ADA	CHAR (10)	NOTE, REFMAN, TEAM, MEMORY, NTEAM, OTHER	P66, D60
CH_ERR_GEN		TABLE CONTAINING ERROR CHARACTERISTICS FOR PARTICULAR CHANGES IDENTIFIED AS ERROR CORRECTIONS			
	CHANGE_NO	FORM NUMBER OF CRF	CHAR (6)		P63, D62
	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	P76, 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	CHAR (6)		P63, D62
	ERR_TOOLS	ADA TOOLS USED THAT AIDED IN DETECTION OR CORRECTION OF ERROR	CHAR (10)	COMPI, SYMDEB, LSE, CMS, SCA, PCA, DECTM, OTHER	P67, D61
COMPUTER		TABLE CONTAINING INFORMATION ABOUT COMPUTERS USED ON VARIOUS PROJECTS			
	CPU_NAME	SHORT, UNIQUE NAME IDENTIFYING A PARTICULAR COMPUTER	CHAR (10)		M4 M5
	C_FULL_NAME	COMPUTER FULL NAME	CHAR (20)		
COM_PURPOSE		TABLE CONTAINING PURPOSES REPORTED ON COFs FOR PARTICULAR COMPONENTS			
	COM_NO	ID UNIQUELY IDENTIFYING EACH COMPONENT	NUMBER (7, 0)		
	PURPOSE	MAJOR PURPOSE(S) OF COMPONENT	CHAR (10)	IOPRO, ALCOMP, DATRA, LODEC, CNTROMOD, INTOP, ADAPR, ADADA	P59, D58

5063(A)G-(6)-25

Table 4-1. SEL Database Tables and Views--Table and Column Descriptions (3 of 9)

TABLE OR VIEW NAME	COLUMN NAME	DESCRIPTION	TYPE	VALID CODE/VALUE	REFERENCE ID
COM_SOURCE		TABLE CONTAINING COF INFORMATION FOR ALL COMPONENTS			
	COM_NO	ID UNIQUELY IDENTIFYING EACH COMPONENT	NUMBER (7,0)		
	PROG_ID	ID UNIQUELY IDENTIFYING EACH PROGRAMMER	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	
	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, 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, D52
COM_STAT		TABLE CONTAINING COMPONENT STATISTICS FOR ALL COMPONENTS			
	COM_NO	ID UNIQUELY IDENTIFYING EACH COMPONENT	NUMBER (7, 0)		
	C_LINE	TOTAL NUMBER OF LINES OF CODE (WITH COMMENTS) IN COMPONENT	NUMBER (6, 0)		P155
	C_EXE_S	TOTAL NUMBER OF EXECUTABLE SOURCE CODE STATEMENTS IN COMPONENT	NUMBER (6, 0)		P154
	C_C_LINE	TOTAL NUMBER OF COMMENT LINES IN COMPONENT	NUMBER (6,0)		P156
EFF_ACT		TABLE CONTAINING PROGRAMMER ACTIVITY HOURS FROM PRFs AND SERVICE PERSONNEL HOURS FROM SPFs FOR ALL PROJECT, PROGRAMMER, AND WEEK COMBINATIONS			
	EFF_ID	VALUES FROM P_ID (EFF_PROJ) OR PS_ID (EFF_SUB)	NUMBER (10, 0)		
	ACTIVITY	ACTIVITY TO WHICH PROGRAMMER OR SERVICE PERSONNEL IS CHARGING TIME ON PRF OR SPF	CHAR (10)	PREDES, CREDES, RDREVDES, WRCODE, RDREVCOD, TSTCODUN, DEBUG, INTTEST, ACCTEST, OTHER, SUPPORT	
	ACT_HR	ACTUAL HOURS SPENT IN PARTICULAR ACTIVITY	NUMBER (10, 2)		P25 TO P34 D23 TO D32 P39 TO P43 D44 TO D48
EFF_FORM		TABLE CONTAINING FORM IDENTIFICATION AND STATUS INFORMATION FOR EACH PROJECT, PROGRAMMER AND WEEK COMBINATION; ENTERED FROM PRFs OR SPFs			

5063G-(5)-26

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

TABLE OR VIEW NAME	COLUMN NAME	DESCRIPTION	TYPE	VALID CODE/VALUE	REFERENCE ID
EFF_FORM (CONT'D)	P_ID	P_ID VALUE FROM TABLE EFF_PROJ	NUMBER (10, 0)		
	FORM_NO	FORM NUMBER OF PRF OR SPF	CHAR (6)	PRF, SPF	D37, D49
	FORM_TYPE	TYPE OF DATA COLLECTION FORM	CHAR (6)		
	STATUS	STATUS OF PRF OR SPF	CHAR (10)	UNCHK, HCCORRECT, HCERROR, VERAP	
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 PROGRAMMER	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF PRF OR SPF	DATE		P23, D22
	PROG_ID	ID UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (5, 0)		
	P_ID	SURROGATE KEY REPRESENTING UNIQUE PROJ_NO, PROG_ID, AND SUB_DATE COMBINATION	NUMBER (10, 0)		
EFF_SUB		TABLE ASSOCIATING P_ID FROM EFF_PROJ 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	CHAR (5)		P47, D51, D152
	PS_ID	SURROGATE KEY REPRESENTING UNIQUE P_ID AND SUB_PRE COMBINATION	NUMBER (10, 0)		
EFF_SUPER		TABLE CONTAINING PERCENTAGE OF TIME SPENT DOING SUPERVISORY WORK FOR A PARTICULAR PROJECT, PROGRAMMER, AND WEEK COMBINATION			
	P_ID	P_ID VALUE FROM TABLE EFF_PROJ	NUMBER (10, 0)		
	PER_SUP	PERCENTAGE OF SUPERVISORY TIME FOR THIS PROGRAMMER, PROJECT, AND WEEK	NUMBER (6, 2)		
PERSONNEL		TABLE CONTAINING INFORMATION ABOUT PERSONNEL FOR WHOM HOURS ARE RECORDED ON VARIOUS PROJECTS			
	PROG_ID	ID UNIQUELY IDENTIFYING EACH PROGRAMMER	NUMBER (5, 0)		
	FORM_NAME	PROGRAMMER NAME AS IT APPEARS ON VARIOUS FORMS	CHAR (15)	THIS FIELD ALSO INCLUDES THE FOLLOWING "SERVICES" PROGRAMMER NAMES LIBARIAN - LIBRARIANS OTHSUPP - OTHER SUPPORT PERSONNEL PROGMGMT - PROGRAM MANAGEMENT PERSONNEL SECRTARY - SECRETARIES TECHPUBS - TECHNICAL PUBLICATIONS PERSONNEL	M1, P24, D21, P55, D50, P64 D61

5063(A)G-(6)-27

Table 4-1. SEL Database Tables and Views--Table and Column Descriptions (5 of 9)

TABLE OR VIEW NAME	COLUMN NAME	DESCRIPTION	TYPE	VALID CODE/VALUE	REFERENCE ID
PERSONNEL (CONT'D)	FULL_NAME	FULL DESCRIPTIVE NAME OF PROGRAMMER	CHAR (30)		M2
	DATE_ENTRY	DATE ON WHICH PROGRAMMER WAS ENTERED INTO SYSTEM	DATE		M3
PROJECT		TABLE CONTAINING INFORMATION ABOUT ALL PROJECTS IN THE DATABASE			
	PROJ_NAME	PROJECT NAME	CHAR (8)		P1, D1
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (3, 0)		
	PROJ_TYPE	PROJECT CATEGORY	CHAR (10)	ATTITUDE, AGSS, SIM, ORBIT, SCIENTIFIC, DATABASE, REALTIME, TOOL, OTHER	P2
	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	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF PCSF	DATE		P124, D83
	CPU_NAME	SHORT NAME IDENTIFYING COMPUTER USED ON PROJECT (FROM COMPUTER TABLE)	CHAR(10)		P134, D38
	TOTAL_HRS	TOTAL COMPUTER HOURS USED FOR PARTICULAR COMPUTER ON PROJECT	NUMBER (10, 2)		P135, D94
	T_RUN	TOTAL NUMBER OF RUNS FOR PARTICULAR COMPUTER ON PROJECT	NUMBER (6, 0)		P136, D95
PROJ_EST		TABLE CONTAINING ESTIMATED STATISTICS FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (3,0)		P13, D2
	SUB_DATE	SUBMISSION DATE OF PEF	DATE		P14, D14
	T_SYS	ESTIMATED TOTAL NUMBER OF SUBSYSTEMS	NUMBER (4, 0)		P15, D15
	T_COM	ESTIMATED TOTAL NUMBER OF COMPONENTS	NUMBER (4, 0)		P16, D16
	T_LINE	ESTIMATED TOTAL NUMBER OF LINES OF CODE	NUMBER (7, 0)		P19, D17
	T_NEW_LINE	ESTIMATED TOTAL NUMBER OF NEW LINES OF CODE	NUMBER (6, 0)		P19, D17
	T_MOD_LINE	ESTIMATED TOTAL NUMBER OF MODIFIED LINES OF CODE	NUMBER (6, 0)		P18, D18
	T_OLD_LINE	ESTIMATED TOTAL NUMBER OF OLD LINES OF CODE	NUMBER (6, 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

5063(A)G-(6)-28

Table 4-1. SEL Database Tables and Views--Table and Column Descriptions (6 of 9)

TABLE OR VIEW NAME	COLUMN NAME	DESCRIPTION	TYPE	VALID CODE/VALUE	REFERENCE ID
PROJ_EST (CONT'D)	SER_HR	ESTIMATED TOTAL SERVICES HOURS	NUMBER (10, 2)		P23, D13
PROJ_EST_PHASE		TABLE CONTAINING ESTIMATED AND AT-COMPLETION PHASE DATES FOR ALL PROJECTS IN THE DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF PEF OR PCSF	DATE		P5, D2, P124, D83
	PHASE_CO	PHASE CODE IDENTIFYING DIFFERENT PHASES IN LIFE OF PROJECT	CHAR (10)	REQNT, DESGN, CODET, SYSTE, ACCTE, CLEAN, MAINT	
	START_DATE	START DATE OF A PARTICULAR PHASE	DATE		D3 TO D10, D84 TO D91, P6 TO P12, P125 TO P131
	END_DATE	END DATE OF A PARTICULAR PHASE	DATE		D3 TO D10, D84 TO D91, P6 TO P12, P125 TO P131
PROJ_FORM		TABLE CONTAINING FORM IDENTIFICATION AND STATUS INFORMATION FOR PEF, PCSF, SEF, AND SPF DATA			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF SPF, PEF, PCSF, OR SEF	DATE		D83, D22, D2
	FORM_NO	FORM NUMBER OF SPF, PEF, PCSF, OR SEF	CHAR (6)	SPF, PEF, PCSF, SEF	D150, D20, D49, D113
	FORM_TYPE	TYPE OF DATA COLLECTION FORM	CHAR (6)		
	STATUS	STATUS CODE FOR FORM DATA	CHAR (10)	UNCHK, HCCORRECT, HCERROR, VERAP	
PROJ_GRH		TABLE CONTAINING GROWTH HISTORY INFORMATION FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF SPF	DATE		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_MESS		TABLE CONTAINING GENERAL PROJECT DESCRIPTION INFORMATION FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (3, 0)		
	MESS_TYPE	GENERAL PROJECT DESCRIPTION CODES	CHAR (10)	COMPACC, CONLIB, CSCP, CURPH, DEVMA, GHTOOL, GSFCP, SELF, TASKNO, TEXT1, TEXT2, TEXT3, TEXT4, TEXT5, TEXT6, TEXT7, TEXT8, TEXT9, TEXT10	

5063G-(5)-29

Table 4-1. SEL Database Tables and Views--Table and Column Descriptions (7 of 9)

TABLE OR VIEW NAME	COLUMN NAME	DESCRIPTION	TYPE	VALID CODE/VALUE	REFERENCE ID
PROJ_MESS (CONT'D)	MESSAGE	GENERAL PROJECT DESCRIPTION	CHAR (65)		P4
	DATE_ENTRY	ENTRY DATE OF EACH MESSAGE	DATE		
PROJ_PROD		TABLE CONTAINING WEEKLY COMPUTER RESOURCE USE INFORMATION FOR ALL PROJECTS IN DATABASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF SPF	DATE		P23, D22
	RES_NAME	SHORT NAME IDENTIFYING COMPUTER USED ON A PROJECT (FROM COMPUTER TABLE)	CHAR (10)		P44, D38
	RES_HR	TOTAL CPU HOURS USED IN CURRENT WEEK	NUMBER (10, 2)		P45, D39
	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	NUMBER (3, 0)		
	EVALUATE	INTEGER INDICATING THE VALUE OF PARTICULAR MEAS_TYPE	NUMBER (1, 0)	1 TO 5	P68 TO P107 P09 TO P123
	MEAS_TYPE	CODES IDENTIFYING PROJECT SUBJECTIVE CHARACTERISTICS	CHAR (10)	PM01, PM02, PM03, PM04, PM05, PM06, ST07, ST08, ST09, ST10, TM11, TM12, TM13, TM14, TM15, PC16, PC17, PC18, PC19, PC20, PC21, PC22, PC23, PC24, EN25, EN26, EN27, EN28, EN29, EN30, PT31, PT32, PT33, PT34, PT35, PT36	
PROJ_SEF_SEC		TABLE CONTAINING SECONDARY-LEVEL INFO, AS RECORDED ON SEFs, FOR ALL PROJECTS IN DATA BASE			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (3, 0)		
	MEAS_TYPE	CODE IDENTIFYING PROJECT CHARACTERISTICS AND TOOLS USED	CHAR (10)	PC21	
	SECOND_L	SECONDARY LEVEL INFORMATION FOR PARTICULAR MEAS_TYPE. AT PRESENT, ALL THE CODES STORED HERE ARE FOR "USE OF TOOLS" (PC21)	CHAR (10)	COMPI, 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	NUMBER (3, 0)		
	SUB_DATE	SUBMISSION DATE OF PCSF	DATE		P124, D83
	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

5063G-(6)-30

Table 4-1. SEL Database Tables and Views--Table and Column Descriptions (8 of 9)

TABLE OR VIEW NAME	COLUMN NAME	DESCRIPTION	TYPE	VALID CODE/VALUE	REFERENCE ID
PROJ_STAT (CONT'D)	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 NUMBER OF LINES OF CODE	NUMBER (7, 0)		P141, D100
	T_NEW_LINE	TOTAL NUMBER OF NEW LINES OF CODE	NUMBER (6, 0)		P142, D101
	T_MOD_LINE	TOTAL NUMBER OF MODIFIED LINES OF CODE	NUMBER (6, 0)		P143, D102
	T_OLD_LINE	TOTAL NUMBER OF OLD LINES OF CODE	NUMBER (6, 0)		P144, D103
	T_COMMENT	TOTAL NUMBER OF COMMENT STATEMENTS	NUMBER (6, 0)		P145, D104
	T_EXE_MOD	TOTAL NUMBER OF EXECUTABLE MODULES	NUMBER (4, 0)		P146, D105
	T_NEW_MOD	TOTAL NUMBER OF NEW MODULES	NUMBER (4, 0)		P147, D106
	T_MOD_MOD	TOTAL NUMBER OF MODIFIED MODULES	NUMBER (4, 0)		P148, D107
	T_OLD_MOD	TOTAL NUMBER OF OLD MODULES	NUMBER (4, 0)		P149, D108
	T_EXE_STAT	TOTAL NUMBER OF EXECUTABLE STATEMENTS	NUMBER (6, 0)		P150, D109
	T_NEW_STAT	TOTAL NUMBER OF NEW EXECUTABLE STATEMENTS	NUMBER (6, 0)		P151, D110
	T_MOD_STAT	TOTAL NUMBER OF MODIFIED EXECUTABLE STATEMENTS	NUMBER (6, 0)		P152, D111
	T_OLD_STAT	TOTAL NUMBER OF OLD EXECUTABLE STATEMENTS	NUMBER (6, 0)		P153, D112
PROJ_SUB		TABLE ASSOCIATING PROJECT AND SUBSYSTEM WITH SURROGATE KEY THAT UNIQUELY IDENTIFIES THE SUB-SYSTEM FOR USE IN OTHER TABLES			
	PROJ_NO	ID UNIQUELY IDENTIFYING EACH PROJECT	NUMBER (3, 0)		
	SUB_PRE	SUBSYSTEM PREFIX	CHAR (5)		P47, D51, D152
	SUBSY_ID	SURROGATE KEY REPRESENTING UNIQUE PROJ_NO AND SUB_PRE COMBINATION	NUMBER (5, 0)		
	SUB_DATE	DATE SUBSYSTEM WAS ENTERED	DATE		P50, D151
SPECIAL_ACT		TABLE CONTAINING PROGRAMMER ACTIVITY HOURS FROM PRFs (PART C) FOR ALL PROJECT, PROGRAMMER, AND WEEK COMBINATIONS		REWORK, ENHANCE, DOCUMENT, REUSE	
	EFF_ID	VALUES FROM P_ID (EFF_PROJ) OR PS_ID (EFF_SUB)	NUMBER (10, 0)		
	SP_ACTIVITY	SPECIAL ACTIVITY TO WHICH PROGRAMMER IS CHARGING TIME ON PRF	CHAR (10)		
	ACT_HR	ACTUAL HOURS SPENT IN A PARTICULAR ACTIVITY	NUMBER (10, 2)		P35 TO P38, D33 TO D36
SUBSYSTEM		TABLE CONTAINING INFORMATION FOR PARTICULAR SUBSYSTEMS, AS RECORDED ON SIFs		USERINT, DPDC, REALTIME, GRAPH, CPEXEC, SYSSERV, MATHCOMP	
	SUBSY_ID	ID UNIQUELY IDENTIFYING EACH SUBSYSTEM	NUMBER (5, 0)		
	NAME	SUBSYSTEM DESCRIPTIVE NAME	CHAR (40)		P48, D153

5063G-(6)-31

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

TABLE OR VIEW NAME	COLUMN NAME	DESCRIPTION	TYPE	VALID CODE/VALUE	REFERENCE ID
SUBSYSTEM (CONT'D)	FUNCTION	SPECIFIC FUNCTION THAT SUBSYSTEM PERFORMS	CHAR (10)		P49, D154
SUB_COM		TABLE ASSOCIATING SUBSYSTEM AND COMPONENT NAME WITH SURROGATE KEY THAT UNIQUELY IDENTIFIES THE COMPONENT FOR USE IN OTHER TABLES			
	SUBSY_ID	ID UNIQUELY IDENTIFYING EACH SUBSYSTEM	NUMBER (5, 0)		
	COM_NAME	COMPONENT DESCRIPTIVE NAME	CHAR (40)		P51, D53
	COM_NO	SURROGATE KEY REPRESENTING UNIQUE SUBSY_ID AND COM_NAME COMBINATION	NUMBER (7, 0)		
	COM_DATE	DATE ON WHICH COMPONENT IS ENTERED INTO DATABASE	DATE		P52
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)		
	CODE	ABBREVIATED CODE	CHAR (10)		
	VALUE	FULL DESCRIPTION OF CODE	CHAR (75)		
V_PROJ_COM		VIEW THAT JOINS THE PROJECT, PROJ_SUB, AND SUB_COM TABLES			
	PROJ_NAME	SAME AS PROJ_NAME IN PROJECT	CHAR		
	SUB_PRE	SAME AS SUB_PRE IN PROJ_SUB	CHAR		
	COM_NAME	SAME AS COM_NAME IN SUB_COM	CHAR		
	COM_NO	SAME AS COM_NO IN SUB_COM	NUMBER		
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		
	SUB_PRE	SAME AS SUB_PRE IN EFF_SUB	CHAR		
	ACTIVITY	SAME AS ACTIVITY IN EFF_ACT	CHAR		
	ACT_HR	SAME AS ACT_HR IN EFF-ACT	NUMBER		
V_SUBSYSTEM_INFO		VIEW THAT JOINS THE PROJECT, PROJ_SUB, AND SUBSYSTEM TABLES			
	PROJ_NAME	SAME AS PROJ_NAME IN PROJECT	CHAR		
	SUB_PRE	SAME AS SUB_PRE IN PROJ-SUB	CHAR		
	NAME	SAME AS NAME AS IN SUBSYSTEM	CHAR		
	FUNCTION	SAME AS FUNCTION IN SUBSYSTEM	CHAR		
	SUB_DATE	SAME AS SUB_DATE IN PROJECT	DATE		

*NOTE: SEE APPENDIX A FOR A DESCRIPTION OF ALL CODES AND VALUES.

5063G-(6)-32

Table 4-2. SEL Database Tables and Views--Technical Specifications (1 of 10)

TABLE OR VIEW NAME	COLUMN NAME	TYPE	WIDTH	KEY[1]	NULLS[3]	INDEXED[2]	CLUSTERED	UNDERLYING TABLE NAME
AUTHORIZE	ACCESS_TYPE	CHAR	10		N. NULL			USER_CLASS_ACCESS
	ORA_USER_ID	CHAR	20		N. NULL			USER_CLASS
CHANGE	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX		
	CH_TYPE	CHAR	10		NULL	INDEX		
	DATE_COMP	DATE	9		NULL			
	DATE_DETER	DATE	9		NULL			
	EFF_ADA	CHAR	1		NULL			
	EFF_COM_CH	CHAR	10		NULL			
	EFF_ISO_CH	CHAR	10		NULL			
	EFF_ONE	CHAR	1		NULL			
	EFF_OTHER	CHAR	1		NULL			
	EFF_PARPA	CHAR	1		NULL			
	FORM_TYPE	CHAR	6		N. NULL			
	NUM_COM_CH	NUMBER	2, 0		NULL			
	NUM_COM_EX	NUMBER	2, 0		NULL			
	PROG_ID	NUMBER	5, 0		N. NULL	INDEX		
	STATUS	CHAR	10		N. NULL	INDEX		
	SUB_DATE	DATE	9		N. NULL	INDEX		
CHANGE_COM	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX		
	COM_NO	NUMBER	7, 0	PK	N. NULL	U. INDEX		
CH_ADAFEAT	ADA_FEATURE	CHAR	10	PK	N. NULL	U. INDEX		
	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX		
CH_ERR_ARES	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX		
	ERR_ARES	CHAR	10	PK	N. NULL	U. INDEX		
CH_ERR_GEN	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX		
	ERR_ACAUSE	CHAR	10		NULL	INDEX		
	ERR_ADOC	CHAR	1		NULL			
	ERR_CLASS	CHAR	10		NULL			
	ERR_COMIS	CHAR	1		NULL			
	ERR_OMIS	CHAR	1		NULL			
	ERR_SOURCE	CHAR	10		NULL			
	ERR_TYPO	CHAR	1		NULL			

5063(1)-1

[1] PK: PRIMARY KEY
[2] U. INDEX: UNIQUE INDEX
[3] N. NULL = NOT NULL

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

TABLE OR VIEW NAME	COLUMN NAME	TYPE	WIDTH	KEY[1]	NULLS[3]	INDEXED[2]	CLUSTERED	UNDERLYING TABLE NAME
CH_ERR_TOOLS	CHANGE_NO	CHAR	6	PK	N. NULL	U. INDEX		
	ERR_TOOLS	CHAR	10	PK	N. NULL	U. INDEX		
COMPUTER	CPU_NAME	CHAR	10	PK	N. NULL	U. INDEX		
	C_FULL_NAME	CHAR	20		N. NULL			
COM_PURPOSE	COM_NO	NUMBER	7, 0	PK	N. NULL	U. INDEX		
	PURPOSE	CHAR	10	PK	N. NULL	U. INDEX		
COM_SOURCE	COM_NO	NUMBER	7, 0	PK	N. NULL	U. INDEX		
	COM_TYPE	CHAR	10		NULL			
	CREATE_DATE	DATE	9		NULL	INDEX		
	DIFFICULTY	NUMBER	2, 0		NULL			
	FORM_NO	CHAR	6		N. NULL	U. INDEX		
	FORM_TYPE	CHAR	6		N. NULL			
	ORI_TYPE	CHAR	10		NULL			
	PROG_ID	NUMBER	5, 0		NULL			
	STATUS	CHAR	10		N. NULL	INDEX		
	SUB_DATE	DATE	9		NULL	INDEX		
COM_STAT	COM_NO	NUMBER	7, 0	PK	N. NULL	U. INDEX		
	C_C_LINE	NUMBER	6, 0		NULL			
	C_EXE_S	NUMBER	6, 0		NULL			
	C_LINE	NUMBER	6, 0		NULL			
CRF_TEMP_CHANGE_COM	USER_ID	NUMBER		PK	N. NULL			
	SUB_PRE	CHAR	5	PK	N. NULL			
	COM_NAME	CHAR	40	PK	N. NULL	U. INDEX		
	COM_NO	NUMBER	7		N. NULL			
						U. INDEX		
DUMMY	HIDDEN	CHAR	1		NULL			
EFF_ACT	ACTIVITY	CHAR	10	PK	N. NULL			
	ACT_HR	NUMBER	10, 2		N. NULL	U. INDEX		
	EFF_ID	NUMBER	10, 0	PK	N. NULL	INDEX		
EFF_FORM	FORM_NO	CHAR	6		N. NULL	U. INDEX		
	FORM_TYPE	CHAR	6		N. NULL	U. INDEX		
	P_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX		
	STATUS	CHAR	10		N. NULL	U. INDEX		

5063(1)-2

[1] PK: PRIMARY KEY
[2] U. INDEX: UNIQUE INDEX
[3] N. NULL = NOT NULL

Table 4-2. SEL Database Tables and Views--Technical Specifications (3 of 10)

TABLE OR VIEW NAME	COLUMN NAME	TYPE	WIDTH	KEY[1]	NULLS[3]	INDEXED[2]	CLUSTERED	UNDERLYING TABLE NAME
EFF_PROJ	PROG_ID	NUMBER	5, 0	PK	N. NULL			
	PROJ_NO	NUMBER	3, 0	PK	N. NULL			
	P_ID	NUMBER	10, 0		N. NULL			
	SUB_DATE	DATE	9	PK	N. NULL			
EFF_SUB	PS_ID	NUMBER	10, 0		N. NULL	U. INDEX		
	P_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX		
	SUB_PRE	CHAR	5	PK	N. NULL	U. INDEX		
EFF_SUPER	PER_SUP	NUMBER	6, 2		N. NULL			
	P_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX		
GENERATE_SAT_DAY	SAT_DAY	DATE	9	PK	N. NULL	U. INDEX		
	SCRIPT_NO	NUMBER	10, 0	PK	N. NULL	U. INDEX		
PERM_SCRIPT	ORA_USER	CHAR	20	PK	N. NULL	U. INDEX		
	OUT_FILE	CHAR	20		NULL			
	OUT_ROUTING	CHAR	20		N. NULL			
	SCRIPT_NAME	CHAR	20	PK	N. NULL	U. INDEX		
	SCRIPT_NO	NUMBER	10, 0		N. NULL	U. INDEX		
PERSONNEL	DATE_ENTRY	DATE	9		N. NULL			
	FORM_NAME	CHAR	15		N. NULL	U. INDEX		
	FULL_NAME	CHAR	30		NULL			
	PROG_ID	NUMBER	5, 0	PK	N. NULL	U. INDEX		
PROJECT	ACTIVE_STATUS	CHAR	10		NULL		PROJ_SUB	
	PROJ_NAME	CHAR	8	PK	N. NULL	U. INDEX		
	PROJ_NO	NUMBER	3, 0		N. NULL	U. INDEX		
	PROJ_TYPE	CHAR	10		NULL			
PROJ_CPU_STAT	CPU_NAME	CHAR	10	PK	N. NULL	U. INDEX		
	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX		
	TOTAL_HRS	NUMBER	10, 2		NULL			
	T_RUN	NUMBER	6, 0		NULL			

5063(1)-3

[1] PK: PRIMARY KEY
[2] U. INDEX: UNIQUE INDEX
[3] N. NULL = NOT NULL

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

TABLE OR VIEW NAME	COLUMN NAME	TYPE	WIDTH	KEY[1]	NULLS[3]	INDEXED[2]	CLUSTERED	UNDERLYING TABLE NAME
PROJ_EST	MAN_HR	NUMBER	10, 2		NULL			
	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	PRO_HR	NUMBER	10, 2		NULL			
	SER_HR	NUMBER	10, 2	PK	NULL	U. INDEX		
	SUB_DATE	DATE	9		N. NULL			
	T_COM	NUMBER	4, 0		NULL			
	T_LINE	NUMBER	7, 0		NULL			
	T_MOD_LINE	NUMBER	6, 0		NULL			
	T_NEW_LINE	NUMBER	6, 0		NULL			
	T_OLD_LINE	NUMBER	6, 0		NULL			
	T_SYS	NUMBER	4, 0		NULL			
	END_DATE	DATE	9		NULL			
PROJ_EST_PHASE	PHASE_CO	CHAR	10	PK	N. NULL	U. INDEX		
	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	START_DATE	DATE	9		N. NULL			
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX		
	FORM_NO	CHAR	6	PK	N. NULL	U. INDEX		
PROJ_FORM	FORM_TYPE	CHAR	6	PK	N. NULL	U. INDEX		
	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	STATUS	CHAR	10		N. NULL	INDEX		
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX		
	GR_CH	NUMBER	6, 0		NULL			
PROJ_GRH	GR_LINE	NUMBER	7, 0		NULL			
	GR_MOD	NUMBER	4, 0		NULL			
	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX		
	DATE_ENTRY	DATE	9		N. NULL			
PROJ_MESS	MESSAGE	CHAR	65		N. NULL			
	MESS_TYPE	CHAR	10	PK	N. NULL	U. INDEX		
	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX		
PROJ_PROD	RES_HR	NUMBER	10, 2		NULL			
	RES_NAME	CHAR	10	PK	N. NULL	U. INDEX		
	RES_RUN	NUMBER	5, 0		NULL			
	SUB_DATE	DATE	9	PK	N. NULL	U. INDEX		

5063(1)-4

[1] PK: PRIMARY KEY
[2] U. INDEX: UNIQUE INDEX
[3] N. NULL = NOT NULL

Table 4-2. SEL Database Tables and Views--Technical Specifications (5 of 10)

TABLE OR VIEW NAME	COLUMN NAME	TYPE	WIDTH	KEY[1]	NULLS[3]	INDEXED[2]	CLUSTERED	UNDERLYING TABLE NAME
PROJ_SEF	EVALUATE	NUMBER	1, 0		NULL			
	MEAS_TYPE	CHAR	10	PK	N. NULL	U. INDEX		
	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX		
PROJ_SEF_SEC	MEAS_TYPE	CHAR	10	PK	N. NULL	U. INDEX		
	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	SECOND_L	CHAR	10	PK	N. NULL	U. INDEX		
PROJ_STAT	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	SER_HR	NUMBER	10, 2		NULL			
	SUB_DATE	DATE	9		N. NULL			
	TECH_MAN_HR	NUMBER	10, 2		NULL			
	T_CH	NUMBER	6, 0		NULL			
	T_COM	NUMBER	4, 0		NULL			
	T_COMMENT	NUMBER	6, 0		NULL			
	T_DOC	NUMBER	6, 0		NULL			
	T_EXE_MOD	NUMBER	4, 0		NULL			
	T_EXE_STAT	NUMBER	6, 0		NULL			
	T_LINE	NUMBER	7, 0		NULL			
	T_MOD_LINE	NUMBER	6, 0		NULL			
	T_MOD_MOD	NUMBER	4, 0		NULL			
	T_MOD_STAT	NUMBER	6, 0		NULL			
	T_NEW_LINE	NUMBER	6, 0		NULL			
	T_NEW_MOD	NUMBER	4, 0		NULL			
	T_NEW_STAT	NUMBER	6, 0		NULL			
	T_OLD_LINE	NUMBER	6, 0		NULL			
	T_OLD_MOD	NUMBER	4, 0		NULL			
	T_OLD_STAT	NUMBER	6, 0		NULL			
	T_SYS	NUMBER	4, 0		NULL			
PROJ_SUB	PROJ_NO	NUMBER	3, 0	PK	N. NULL	U. INDEX	PROJECT	
	SUBSY_ID	NUMBER	5, 0		N. NULL	U. INDEX		
	SUB_DATE	DATE	9		N. NULL			
	SUB_PRE	CHAR	5	PK	N. NULL	U. INDEX		
REP_CODES	CODE	CHAR	10	PK	N. NULL			
	VALUES	CHAR	30		N. NULL			
	FUNCTION	CHAR	15		N. NULL			

5063(1)-5

[1] PK: PRIMARY KEY
[2] U. INDEX: UNIQUE INDEX
[3] N. NULL = NOT NULL

Table 4-2. SEL Database Tables and Views--Technical Specifications (6 of 10)

TABLE OR VIEW NAME	COLUMN NAME	TYPE	WIDTH	KEY[1]	NULLS[3]	INDEXED[2]	CLUSTERED	UNDERLYING TABLE NAME
REP_CONDITIONS	END_DATE	DATE	9		NULL			
	LINES_OF_CODE	NUMBER	5, 0		NULL			
	NUM_COM	NUMBER	5, 0		NULL			
	PROJ_TYPE	CHAR	10		NULL			
	REPORT_SEQ	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	SCRIPT_NO	NUMBER	10, 0	PK	N. NULL	U. INDEX		
	START_DATE	DATE	9		NULL			
SCRIPT_PROJECTS	PROJ_NAME	CHAR	8	PK	N. NULL	U. INDEX		
	REPORT_SEQ	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	SCRIPT_NO	NUMBER	10, 0	PK	N. NULL	U. INDEX		
SCRIPT_REPORT	REPORT_CODE	CHAR	10		N. NULL			
	REPORT_SEQ	NUMBER	3, 0	PK	N. NULL	U. INDEX		
	REPORT_TYPE	CHAR	20		N. NULL			
	REPORT_TYPE_SELECTION	CHAR	10		NULL			
	SCRIPT_NO	NUMBER	10, 0	PK	N. NULL	U. INDEX		
SEQNO	FIELD_NAME	CHAR	30	PK	N. NULL	U. INDEX		
	MAXSEQNO	NUMBER	10, 0		N. NULL			
	TABLE_NAME	CHAR	30	PK	N. NULL	U. INDEX		
SPECIAL_ACT	ACT_HR	NUMBER	10, 2		N. NULL			
	EFF_ID	NUMBER	10, 0	PK	N. NULL	U. INDEX		
	SP_ACTIVITY	CHAR	10	PK	N. NULL	U. INDEX		
SUBSYSTEM	FUNCTION	CHAR	10		NULL			
	NAME	CHAR	40		N. NULL			
	SUBSY_ID	NUMBER	5, 0	PK	N. NULL	U. INDEX		
SUB_COM	COM_DATE	DATE	9		N. NULL			
	COM_NAME	CHAR	40	PK	N. NULL	U. INDEX		
	COM_NO	NUMBER	7, 0		N. NULL	U. INDEX		
	SUBSY_ID	NUMBER	5, 0	PK	N. NULL	U. INDEX		

5063(1)-6

[1] PK: PRIMARY KEY
[2] U. INDEX: UNIQUE INDEX
[3] N. NULL = NOT NULL

Table 4-2. SEL Database Tables and Views--Technical Specifications (7 of 10)

TABLE OR VIEW NAME	COLUMN NAME	TYPE	WIDTH	KEY[1]	NULLS[3]	INDEXED[2]	CLUSTERED	UNDERLYING TABLE NAME
TABLE_PRIVILEGE	ALTER_PRIV	CHAR	1		NULL			
	DELETE_PRIV	CHAR	1		NULL			
	INDEX_PRIV	CHAR	1		NULL			
	INSERT_PRIV	CHAR	1		NULL			
	SELECT_PRIV	CHAR	1		NULL			
	TABLE_NAME	CHAR	40	PK	N. NULL	U. INDEX		
	UPDATE_PRIV	CHAR	1		NULL			
	USER_CLASS	CHAR	20	PK	N. NULL	U. INDEX		
TEMP_ACTIVITY	SAT_DAY	DATE	9					
	ACTIVITY	CHAR	10					
	HOURS	NUMBER	10, 2					
	PROJ_NO	NUMBER	3		N. NULL			
	SUB_HR	NUMBER	10, 2					
	FLAG	CHAR	4					
	SCRIPT_NO	NUMBER	10					
TEMP_FORMCT	SAT_DAY	DATE	9					
	PROJ_NO	NUMBER	3					
	PROG_ID	NUMBER	5		N. NULL			
	FORM_TYPE	CHAR	6					
	SCRIPT_NO	NUMBER	10					
TEMP_MANHRS	FORM_NAME	CHAR	15					
	SAT_DAY	DATE	9		N. NULL			
	HOURS	NUMBER	10, 2					
	PROJ_NO	NUMBER	3					
	PROG_ID	NUMBER	5		N. NULL			
	SUB_HR	NUMBER	10, 2					
	FLAG	CHAR	4					
	P_ID	NUMBER	10					
	SCRIPT_NO	NUMBER	10					
TEMP_SCRIPT	DELETE_STATUS	CHAR	10		N. NULL			
	ORA_USER	CHAR	20		N. NULL			
	OUT_FILE	CHAR	20		NULL			
	OUT_ROUTING	CHAR	20		N. NULL			
	PROCESS_ID	CHAR	20		N. NULL			
	RUN_STATUS	CHAR	10		N. NULL			
	SCRIPT_NO	NUMBER	10, 0	PK	N. NULL	U. INDEX		

5063(1)-7

[1] PK: PRIMARY KEY
[2] U. INDEX: UNIQUE INDEX
[3] N. NULL = NOT NULL

Table 4-2. SEL Database Tables and Views--Technical Specifications (8 of 10)

TABLE OR VIEW NAME	COLUMN NAME	TYPE	WIDTH	KEY[1]	NULLS[3]	INDEXED[2]	CLUSTERED	UNDERLYING TABLE NAME
USER_CLASS	ORA_USER_ID	CHAR	20	PK	N. NULL	U. INDEX		
	USER_CLASS	CHAR	20		N. NULL			
USER_CLASS_ACCESS	ACCESS_TYPE	CHAR	10	PK	N. NULL	U. INDEX		
	USER_CLASS	CHAR	20	PK	N. NULL	U. INDEX		
V_PERM_SCRIPT	SCRIPT_NAME	CHAR	30					PERM_SCRIPT
V_REP_CODES_CRITERIA	VALUE	CHAR	30					REP_CODES
VALIDATION	CODE	CHAR	10	PK	N.NULL	U. INDEX		
	F_NAME	CHAR	20	PK	N. NULL	U. INDEX		
	VALUE	CHAR	75		N. NULL			
VAL_ACTIVE_STATUS	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_ACTIVITY	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_ADA_FEATURE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_CH_TYPE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_COM_CH	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_COM_PURPOSE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_COM_TYPE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_ERR_ACAUSE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_ERR_ARES	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION

5063(6)-8

[1] PK: PRIMARY KEY
[2] U. INDEX: UNIQUE INDEX
[3] N. NULL = NOT NULL

Table 4-2. SEL Database Tables and Views--Technical Specifications (9 of 10)

TABLE OR VIEW NAME	COLUMN NAME	TYPE	WIDTH	KEY[1]	NULLS[3]	INDEXED[2]	CLUSTERED	UNDERLYING TABLE NAME
VAL_ERR_CLASS	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_ERR_SOURCE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_ERR_TOOLS	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_ISO_CH	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_MEAS_TYPE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_MESS_TYPE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_ORI_TYPE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_PHASE_CO	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_PROJ_TYPE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_QA_STATUS	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_REPORT_CODE	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_SECOND_L	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_SP_ACTIVITY	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
VAL_STATUS	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION

5063(1)-9

[1] PK: PRIMARY KEY
[2] U. INDEX: UNIQUE INDEX
[3] N. NULL = NOT NULL

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

TABLE OR VIEW NAME	COLUMN NAME	TYPE	WIDTH	KEY[1]	NULLS[3]	INDEXED[2]	CLUSTERED	UNDERLYING TABLE NAME
VAL_S_FUNCTION	CODE	CHAR	10		N. NULL			VALIDATION
	VALUE	CHAR	75		N. NULL			VALIDATION
V_PROJ_COM	COM_NAME	CHAR	40		N. NULL			SUB_COM
	COM_NO	NUMBER	7, 0		N. NULL			SUB_COM
	PROJ_NAME	CHAR	8		N. NULL			PROJECT
	SUB_PRE	CHAR	5		N. NULL			PROJ_SUB
V_PROJ_SUB_ACT	ACTIVITY	CHAR	10		N. NULL			EFF_ACT
	ACT_HR	NUMBER	10, 2		N. NULL			EFF_ACT
	PROJ_NAME	CHAR	8		N. NULL			PROJECT
	SUB_PRE	CHAR	5		N. NULL			EFF_SUB
V_SUBSYSTEM_INFO	FUNCTION	CHAR	10		NULL			SUBSYSTEM
	NAME	CHAR	40		N. NULL			SUBSYSTEM
	PROJ_NAME	CAHR	8		N. NULL			PROJECT
	SUB_DATE	DATE	7		N. NULL			PROJ_SUB
	SUB_PRE	CHAR	5		N. NULL			PROJ_SUB

5063(1)-10

[1] PK: PRIMARY KEY
[2] U. INDEX: UNIQUE INDEX
[3] N. NULL = NOT NULL

containing the project name, project type, and project status is created. A unique project number is also assigned and stored in the same record. The rest of the project data are stored in various tables. The relationship between tables PROJECT and PROJ_SUB is defined through the project number column.

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 system support tables. Figure 4-3 shows all the tables that are related to the tables containing computer, manpower, and services 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 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 understand how the data are logically divided and stored in the database. Observe that the data elements that make up 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 elements. 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

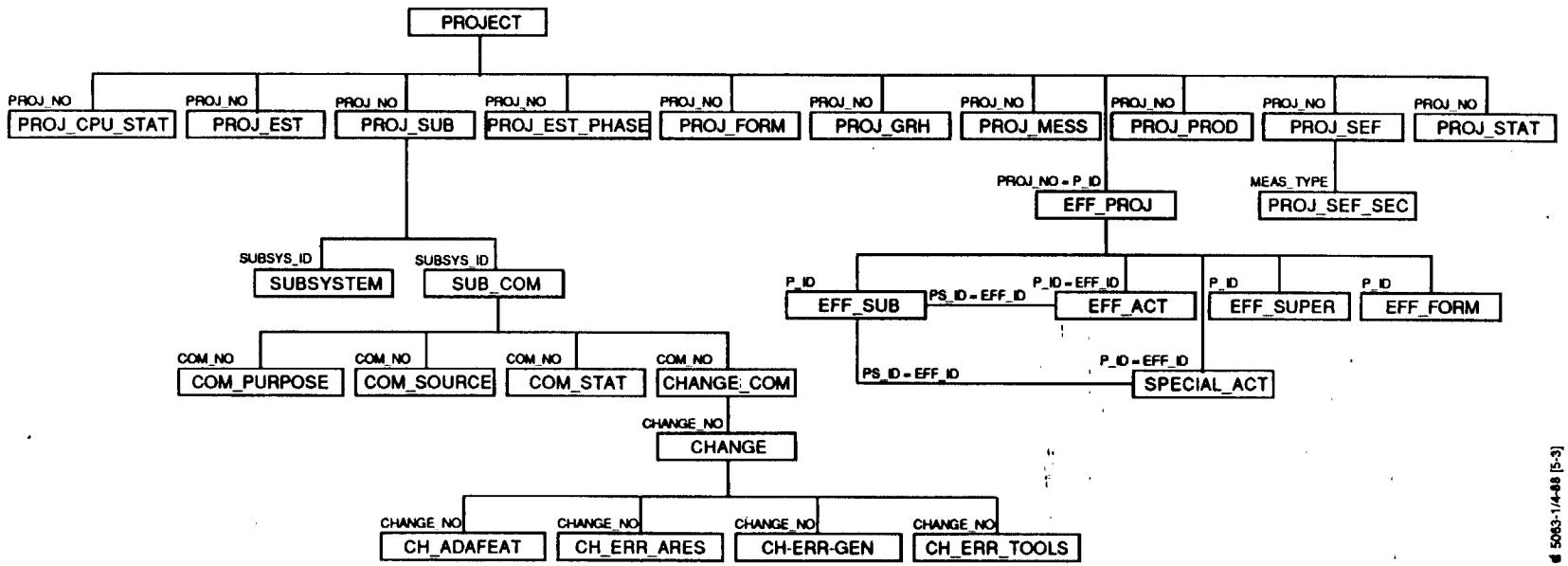


Figure 4-1. Relationships Among Project-Related Tables

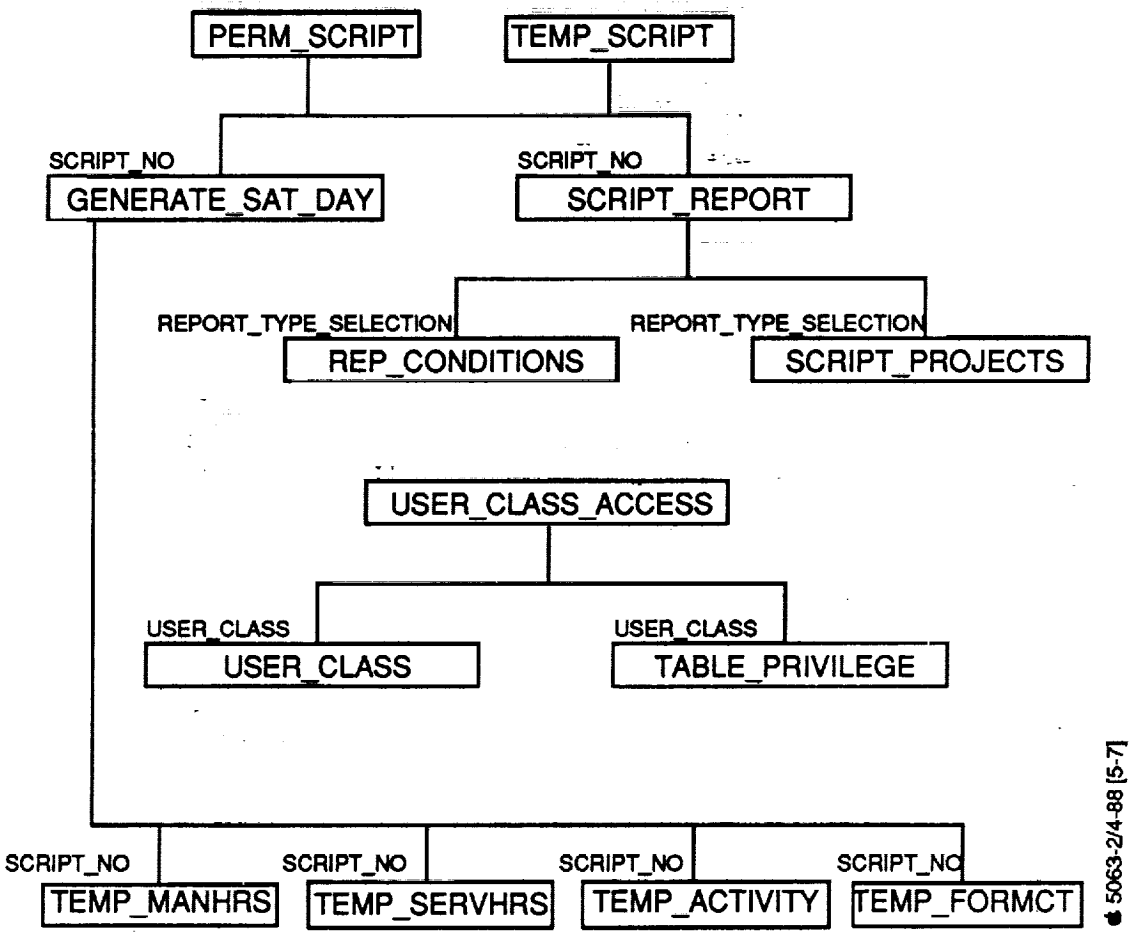


Figure 4-2. Relationships Among Support Data Tables

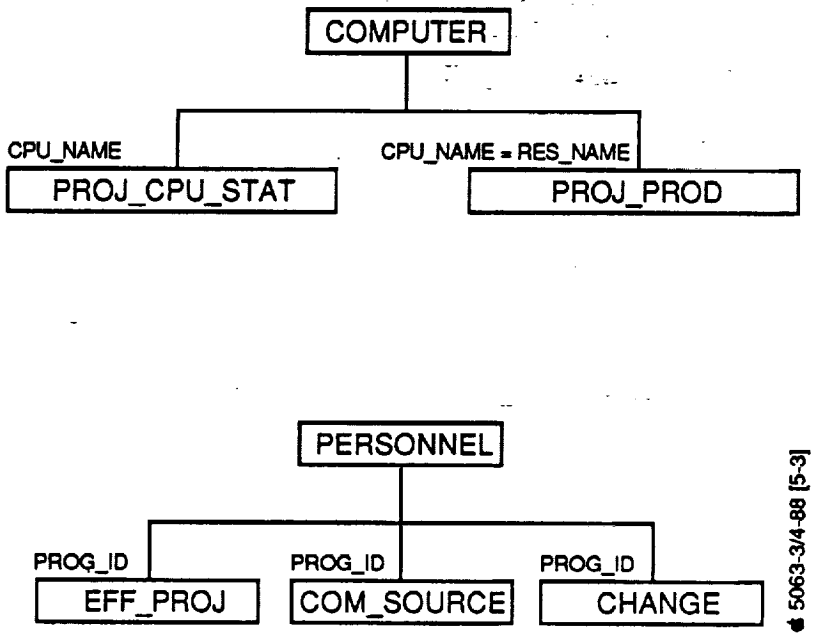


Figure 4-3. Relationships Involving the COMPUTER and PERSONNEL Tables

table because one component can have multiple purposes. This logical partitioning of data is 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 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, some projects may not have subsystem-related activity hours. Thus, 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.

In addition, some of the tables are used as connectors to relate data items together 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, specifically project and subsystem data items that are 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.

CRF_TEMP_CHANGE_COM

This table is used for running the CRF menu screens (CRF_UPDATE, CRF_INSERT, CRF_QA). It contains the component information associated with the current CRF form. The information is uniquely identified with a USER_ID. This is actually the SESSIONID of the current user.

DUMMY

This table is used by the data entry software. 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 database 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.

PERM_SCRIPT

This table is used in generating database reports. It contains header information about the permanent report scripts. A report script is built during interactive report selection via the SEL user interface. The scripts are identified by the script numbers and their owners.

REP_CODES

This table is used as a look up table for the Report Interface System. It contains all of the possible report titles, report types, batch queues, and log printers. For each entry in the table there is a function and a unique code which corresponds to a detailed value. These values have two purposes. They are used to display information in a readable form so that user will easily understand the contents of a report script, and they are used to list available options for queues, printers, etc.

REP_CONDITIONS

This table is used in generating database reports. For each record in table SCRIPT_REPORT that has a value in the field REPORT_TYPE_SELECTION, there will be an entry in this table to further specify the conditions to be applied in selecting a set of projects within that particular report.

SCRIPT_PROJECTS

This table is used in generating database reports. It stores the names of the projects that are selected for a multiple-project report. The only entries stored in this table permanently are for the permanent scripts that have 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.

SCRIPT_REPORT

This table is used in generating database reports. It contains the bodies of all scripts, including both temporary

and permanent scripts. The type of reports within a script, its sequence, and other report-related information are also specified in this table.

SEQNO

This table is used by the data entry software. It contains the maximum values of all the system-generated IDs in the database. The system-generated IDs are used in the following tables and columns:

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

TABLE_PRIVILEGE

This table is used in enrolling database 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 Programmer Activity Hours Reports. It contains all of the possible activities for each week the project has been in a working phase. For each activity and week, the total number of hours worked is also 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 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 working phase. For each form type and week, the total number of forms entered is also stored. To populate this table the GENERATE_SAT_DAY table must first be populated with the correct Saturday dates.

TEMP_MANHRS

This table is used for producing the Manpower Hours Reports. It contains all of the programmer names for each week the project has been in a working phase. For each programmer and week, the total number of hours worked is also 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 database 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 the script numbers.

TEMP_SERVHRS

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

USER_CLASS

This table is used in enrolling database 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, SEL database administrator (DBA), and system maintenance user.

USER_CLASS_ACCESS

This table is used in enrolling database users. For each user class specification, the types of functional access permitted are stored in this table. The current valid types of access are form, query, view, backup, delete, distape, general, insert, update, QA, DBA, import, and restore.

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 Name	Field Name
PROJECT	ACTIVE_STATUS
PROJECT	PROJ_TYPE
PROJ_FORM	STATUS
PROJ_EST_PHASE	PHASE_CO
PROJ_MESS	MESS_TYPE
PROJ_SEF	MEAS_TYPE
PROJ_SEF_SEC	SECOND_L
EFF_FORM	STATUS
EFF_ACT	ACTIVITY
SPECIAL_ACT	SP_ACTIVITY
CHANGE	STATUS
CHANGE	EFF_ISO_CH
CHANGE	EFF_COM_CH
CHANGE	CH_TYPE
CH_ADAFEAT	ADA_FEATURE
CH_ERR_ARES	ERR_ARES
CH_ERR_GEN	ERR_SOURCE
CH_ERR_GEN	ERR_CLASS
CH_ERR GEN	ERR_ACAUSE
CH_ERR	ERR_TOOLS
COM_PURPOSE	PURPOSE
COM_SOURCE	STATUS
COM_SOURCE	ORI_TYPE
COM_SOURCE	COM_TYPE
SUBSYSTEM	FUNCTION
SCRIPT_REPORT	REPORT_CODE
REP_CONDITIONS	PROJ_TYPE

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 structured query language (SQL) queries, but also in the operational 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 mentioned 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 of how to get to the desired data. By using this schema, along with the specific instructions on how to access the SQL 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 IDs used in Sections 2 and 3 that identify the data items in the database and gives the names 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, information needed to obtain a particular piece of data, is also provided for each ID. Under the columns "TARGET TABLE" and "TARGET COLUMN" are the field/table where data are being retrieved. For example, to retrieve the activity hours for a particular programmer (see page 7 of Table 4-4, under ACT_HR/EFF_ACT), the project name, the programmer, the project name, the programmer name, and the submission date of the PRF or the form number must be provided before the appropriate activity hours can be retrieved.

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

TABLE	CONSTRAINT
CHANGE	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE. THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW. THE EFFORT TO IMPLEMENT CHANGES CODE (EFF_COM_CH) MUST EXIST IN THE VAL_COM_CH VIEW. THE EFFORT TO ISOLATE CHANGES CODE (EFF_ISO_CH) MUST EXIST IN THE VAL_ISO_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 CRF FORM NUMBER (CHANGE_NO) MUST BE UNIQUE.
CHANGE_COM	THE COMPONENT NUMBER (COM_NO) MUST EXIST IN THE SUB_COM TABLE. THE CRF FORM NUMBER (CHANGE_NO) MUST EXIST IN THE CHANGE TABLE.
CH_ADAFEAT	THE ADA FEATURE CODE (ADA_FEATURE) MUST EXIST IN THE VAL_ADA_FEATURE VIEW. THE CHANGE NUMBER (CHANGE_NO) MUST EXIST IN THE CHANGE TABLE, THE FLAG INDICATING WHETHER THE USE OF ADA CONTRIBUTED TO THE CHANGE (EFF_ADA) IN THE CHANGE TABLE MUST EQUAL 'Y' FOR THAT CHANGE, AND CH_TYPE MUST BE 'ERRCO'.
CH_ERR_ARES	RESOURCE CODE NEEDED TO CORRECT ADA ERROR (ERR_ARES) MUST EXIST IN THE VAL_ERR_ARES VIEW. THE CHANGE 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'.
CH_ERR_GEN	THE CHANGE 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. CAUSE FOR AN ERROR INVOLVING ADA CODE (ERR_ACAUSE) MUST EXIST IN THE VAL_ERR_ACAUSE VIEW. CLASS OF ERROR CODE (ERR_CLASS) MUST EXIST IN THE VAL_ERR_CLASS VIEW.

5062G(1)-12

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

TABLE	CONSTRAINT
CH_ERR_TOOLS	ADA TOOLS AIDED IN THE DETECTION OR CORRECTION OF ERROR CODE (ERR_TOOLS) MUST EXIST IN THE VAL_ERR_TOOLS VIEW.
	THE CHANGE 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 ERR_ADA MUST EQUAL 'Y'.
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 COF NUMBER (FORM_NO) MUST BE UNIQUE WITHIN THIS TABLE.
	THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW.
	THE COMPONENT TYPE CODE (COM_TYPE) MUST EXIST IN THE VAL_COM_TYPE VIEW.
	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE ORIGIN OF A COMPONENT CODE (ORI_TYPE) MUST EXIST IN THE VAL_ORI_TYPE VIEW.
	THE FORM TYPE (FORM_TYPE) MUST EQUAL 'COF'.
COM_STAT	THE COMPONENT NUMBER (COM_NO) MUST EXIST IN THE SUB_COM TABLE.
CRF_TEMP_CHANGE_COM	SUBSYSTEM PREFIX (SUB_PRE) MUST EXIST IN THE PROJ_SUB TABLE.
	COMPONENT NAME (COM_NAME) MUST EXIST IN THE V_PROJ_COM VIEW.
	COMPONENT NUMBER (COM_NO) MUST EXIST IN THE V_PROJ_COM VIEW.
EFF_ACT	THE EFF_ID MUST EXIST EITHER 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 MUST BE EITHER 'PRF' OR 'SPF'.
	THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW.

5062G(1)-13

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

TABLE	CONSTRAINT
EFF_PROJ	PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE.
	THE PROGRAMMER ID (PROG_ID) MUST EXIST IN THE PERSONNEL TABLE.
	THE SUBMISSION DATE (SUB_DATE) MUST BE A VALID FRIDAY DATE.
	THE P_ID MUST BE UNIQUE.
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.
EFF_SUPER	THE P_ID MUST EXIST IN THE EFF_PROJ 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.
PERM_SCRIPT	THE SCRIPT NUMBER (SCRIPT_NO) MUST BE UNIQUE.
	THE ORACLE USER ID (ORA_USER) MUST EXIST IN THE USER_CLASS TABLE.
	THE VALID VALUES FOR FIELD OUT_ROUTING ARE 'P' FOR PRINTER, 'F' FOR FILE.
	THE OUTPUT FILE NAME (OUT_FILE) MUST BE ENTERED IF THE VALUE IN FIELD OUT_ROUTING EQUALS 'F'.
PROJECT	THE PROJECT NUMBER (PROJ_NO) MUST BE UNIQUE.
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_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.

5062G(1)-14

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

TABLE	CONSTRAINT
PROJ_FORM	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE. THE STATUS CODE (STATUS) MUST EXIST IN THE VAL_STATUS VIEW. THE FORM TYPE (FORM_TYPE) MUST EQUAL 'PEF', 'SPF', 'PCSF', OR 'SEF'. THE FORM NUMBER (FORM_NO) MUST BE UNIQUE WITHIN A PARTICULAR FORM TYPE.
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_MESS	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE. THE GENERAL PROJECT DESCRIPTION CODE (MESS_TYPE) MUST EXIST IN THE VAL_MESS_TYPE VIEW.
PROJ_PROD	THE PROJECT NUMBER (PROJ_NO) MUST EXIST IN THE PROJECT TABLE. THE COMPUTER NAME (RES_NAME) MUST EXIST IN THE COMPUTER TABLE. THE SUBMISSION DATE (SUB_DATE) MUST BE A VALID FRIDAY DATE.
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 OF VARIOUS 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 ID (SUBSY_ID) MUST BE UNIQUE.
REP_CONDITIONS	THE SCRIPT NUMBER (SCRIPT_NO) MUST EXIST IN THE SCRIPT_REPORT TABLE, THE REPORT_TYPE_SELECTION FIELD IN THE SCRIPT_REPORT TABLE MUST EQUAL 'SCONDITION', AND THE REPORT_SEQ MUST EXIST IN THE SCRIPT_REPORT TABLE.

5063G(1)-15

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

TABLE	CONSTRAINT
SCRIPT_PROJECTS	THE SCRIPT NUMBER (SCRIPT_NO) MUST EXIST IN THE SCRIPT_REPORT TABLE AND THE REPORT SEQUENCE (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 TYPE OF REPORT CODE (REPORT_TYPE) MUST EQUAL 'S' FOR SINGLE PROJECT REPORT, 'M' FOR MULTIPLE-PROJECT REPORT, OR 'O' FOR MISCELLANEOUS REPORT. IF REPORT_TYPE EQUALS TO 'M', THE VALID VALUES FOR REPORT_TYPE_SELECTION ARE 'ALL', 'ACTIVE', 'INACTIVE', 'SCONDITION', 'LIST'. IF REPORT_TYPE EQUALS TO 'O', THE REPORT_TYPE_SELECTION IS NULL. IF REPORT_TYPE EQUALS TO 'S', THE VALID VALUES FOR REPORT_TYPE_SELECTION IS A VALID PROJECT NAME (PROJ_NAME) IN PROJECT.
SEQNO	THE TABLE NAME (TABLE_NAME) MUST EXIST IN THE DATABASE. THE FIELD NAME (FIELD_NAME) MUST EXIST IN THAT PARTICULAR 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 NUMBER (COM_NO) MUST BE UNIQUE.
TABLE_PRIVILEGE	THE USER CLASS (USER_CLASS) MUST EXIST IN THE USER_CLASS TABLE. THE TABLE NAME (TABLE_NAME) MUST EXIST IN THE DATABASE.
TEMP_SCRIPT	THE SCRIPT NUMBER (SCRIPT_NO) MUST BE UNIQUE. THE ORACLE USER ID (ORA_USER) MUST EXIST IN THE USER_CLASS TABLE. THE VALID VALUES FOR FIELD OUT_ROUTING ARE 'P' FOR PRINTER, 'F' FOR FILE. THE OUTPUT FILE NAME (OUT_FILE) MUST BE ENTERED IF THE VALUE IN FIELD OUT_ROUTING EQUALS 'F'.

5062G(1)-16

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

TABLE	CONSTRAINT
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.
TEMP_ACTIVITY	THE SCRIPT_NO AND SAT_DAY MUST EXIST IN THE GENERATE_SAT_DAY TABLE.
TEMP_FORMCT	THE SCRIPT_NO AND SAT_DAY MUST EXIST IN THE GENERATE_SAT_DAY TABLE.
TEMP_MANHRS	THE SCRIPT_NO AND SAT_DAY MUST EXIST IN THE GENERATE_SAT_DAY TABLE.
TEMP_SERVHRS	THE SCRIPT_NO AND SAT_DAY MUST EXIST IN THE GENERATE_SAT_DAY TABLE.

5063G(1)-17

Under the heading "Access Path," there is a graph-like diagram showing the access path that an 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 always point to either the intermediate or the final target columns or tables. 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 two examples demonstrate how to interpret the access path diagrams. They also show that some of the access paths may retrieve one 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, or multiple records if more than one subsystem exists. Otherwise, it will return null. In the second example, the access path will return only one 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.

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:

```
[PROJ_NAME] → PROJECT
                  ↓ [PROJ_NO]

              PROJ_SUB
                 ↓
              [SUB_PRE]
```

The first line in the access path shows that PROJ_NAME is the qualified field of the PROJECT table. In other words, the value of the field is 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 the common field, PROJ_NO in this case, that is listed next to the arrow. The down arrow under PROJ_SUB points to the target column SUB_PRE of PROJ_SUB, which is 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 a component was entered into the 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] →        PROJECT
                                [PROJ_NO]
                        ↓
[SUB_PRE]   →        PROJ_SUB
                                [SUBSY_ID]
                        ↓
[COM_NAME]  →        SUB_COM
                                [COM_NO]
                        ↓
                    COM_SOURCE
                        ↓
                  [CREATE_DATE]
```

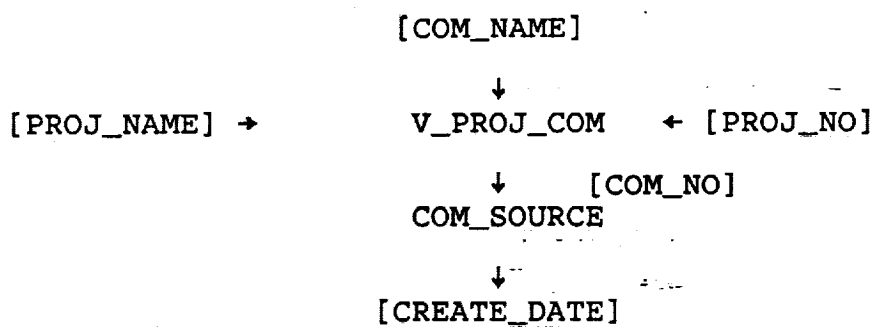
PROJ_NAME, SUB_PRE, and COM_NAME are the qualified fields of tables PROJECT, PROJ_SUB, and SUB_COM, respectively. Tables PROJECT and PROJ_SUB are joined on PROJ_NO; PROJ_SUB and SUB_COM are joined on SUBSY_ID; and SUB_COM and COM_SOURCE are joined on COM_NO. The result is from field CREATE_DATE of the 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 of example 2 to get the same data, i.e., the date 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 joining with the COM_SOURCE table. The result is from 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 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
P85, D77	CH_ADAFEAT	ADA_FEATURE	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] —> CHANGE ↓ [CHANGE_NO] CH_ADAFEAT ↓ [ADA_FEATURE]*CODED FIELD
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	DATE_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, D 76	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

5063-4/2-89[5-7]

Table 4-4. SEL Database Access Paths (2 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
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
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, D60	CHANGE	SUB_DATE	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] —> CHANGE ↓ [SUB_DATE]

5063-5/2-89[5-7]

Table 4-4. SEL Database Access Paths (3 of 18)

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] —> CHANGE ↓ [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] —> CHANGE ↓ [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] —> CHANGE ↓ [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] —> CHANGE ↓ [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] —> CHANGE ↓ [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] —> CHANGE ↓ [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] —> CHANGE ↓ [CHANGE_NO] CH_ERR_GEN ↓ [ERR_SOURCE]* CODED FIELD

5063-6/2-89[5-7]

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

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] —> CHANGE ↓ [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] —> CHANGE ↓ [CHANGE_NO] CH_ERR_TOOLS ↓ [ERR_TOOLS]*CODED FIELD
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
M5	COMPUTER	C_FULL_NAME	COMPUTER SHORT NAME	[CPU_NAME] —> COMPUTER —> [C_FULL_NAME]
M4	COMPUTER	CPU_NAME	NONE	—> COMPUTER —> [CPU_NAME]
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
P53, D54	COM_SOURCE	CREATE_DATE	PROJECT NAME, SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] —>PROJECT ↓ [PROJ_NO] [SUB_PRE] —>PROJ_SUB ↓

5063-7/2-89[5-7]

Table 4-4. SEL Database Access Paths (5 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
P53, D54 (CONT'D)				[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	[PROJ_NAME] —> V_PROJ_COM ↓ [COM_NO] ↓ COM_SOURCE ↓ [FORM_NO]
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, D52	COM_SOURCE	SUB_DATE	PROJECT NAME, SUBSYSTEM PREFIX, AND COMPONENT NAME	[PROJ_NAME] —>PROJECT ↓ [PROJ_NO] ↓ [SUB PRE] —>PROJ_SUB ↓

5063G-(6)-40

Table 4-4. SEL Database Access Paths (6 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
P54, D52 (CONT'D)				SUBSY_ID] [COM_NAME] —> SUB_COM [COM_NO] COM_SOURCE [SUB_DATE]
P156	COM_STAT	C_C_LINE	PROJECT NAME AND COMPONENT NAME	[PROJ_NAME] —> V_PROJ_COM <— [COM_NAME] [PROJ_NO] PROJ_STAT [C_C_LINE]
P154	COM_STAT	C_EXE_S	PROJECT NAME AND COMPONENT NAME	[PROJ_NAME] —> V_PROJ_COM <— [COM_NAME] [COM_NO] COM_STAT [C_EXE_S]
P155	COM_STAT	C_LINE	PROJECT NAME AND COMPONENT NAME	[PROJ_NAME] —> V_PROJ_COM <— [COM_NAME] [PROJ_NO] COM_STAT [C_LINE]
P25, P26, P27, P28, P29, P30, P31, P32, P33, P34, D23 THROUGH D32	EFF_ACT	ACT_HR	PROJECT NAME, PROGRAMMER NAME, WEEK ENDING DATE, AND SUBSYSTEM PREFIX (OPTIONAL)	[PROJ_NAME] —>PROJECT [PROJ_NO] [FORM_NAME] —> PERSONNEL [PROG_ID] —>EFF_PROJ < - [SUB_DATE] [P_ID] —> EFF_SUB <— [SUB_PRE] [ACTIVITY] —> EFF_ACT <——— [PS_ID] [ACT_HR] WHERE ACTIVITY FOR P25, D23 = PREDES ACTIVITY FOR P26, D24 = CREDES ACTIVITY FOR P27, D25 = RDREVCOD ACTIVITY FOR P28, D26 = WRCODE

5063G-(6)-41

Table 4-4. SEL Database Access Paths (7 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
P25, P26, P27, P28, P29, P30, P31, P32, P33, P34, D23 THROUGH D32 (CONT'D)				WHERE ACTIVITY FOR P29, D27 = RDREVDES ACTIVITY FOR P30, D28 = TSTCODUN ACTIVITY FOR P31, D29 = DEBUG ACTIVITY FOR P32, D30 = INTTEST ACTVITY FOR P33, D31 = ACCTEST ACTIVITY FOR P34, D32 = OTHER
P39, P40, P41, P42, P43, D44 TO D48	EFF_ACT	ACT_HR	PROJECT NAME, PROGRAMMER NAME, AND WEEK ENDING DATE	[PROJ_NAME] —>PROJECT [PROJ_NO] [FORM_NAME] —>PERSONNEL [PROG_ID] —> EFF_PROJ <— [SUB_DATE] [P_ID] = [EFF_ID] EFF_ACT [ACT_HR] WHERE FORM_NAME FOR P39, D44 = TECHPUBS FORM_NAME FOR P40, D45 = SECRETARY FORM_NAME FOR P41, D46 = LIBRARIAN FORM_NAME FOR P42, D47 = PROGMGMT FORM_NAME FOR P43, D48 = OTHSUPP
D37, D49	EFF_FORM	FORM_NO	PROJECT NAME AND FORM TYPE	[PROJ_NAME] —> PROJECT [PROJ_NO] EFF_PROJ [P_ID] [FORM_TYPE] —> EFF_FORM [FORM_NO] NOTE: FORM_TYPE FOR D37 = PRF FORM_TYPE FOR D49 = SPF
P23, D22	EFF_PROJ	SUB_DATE		[PROJ_NAME] —>PROJECT [PROJ_NO] EFF_PROJ [SUB_DATE]
M3	PERSONNEL	DATE_ENTRY	PROGRAMMER FORM NAME	[FORM_NAME] —> PERSONNEL —> [DATE_ENTRY]
P24, D21	PERSONNEL	FORM_NAME	PROJECT NAME	[PROJ_NAME] —>PROJECT [PROJ_NO] EFF_PROJ

5063-10/2-89[5-7]

Table 4-4. SEL Database Access Paths (8 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
P24, D21 (CONT'D)				↓ [PROG_ID] —> PERSONNEL ↓ [FORM_NAME] WHERE FORM_NAME ! = TECHPUBS FORM_NAME ! = SECRETARY FORM_NAME ! = LIBRARIAN FORM_NAME ! = PROGMGMT FORM_NAME ! = OTHSUPP
P55, D50	PERSONNEL	FORM_NAME	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]
P64, D61	PERSONNEL	FORM_NAME	CHANGE NUMBER; SEE P63 FOR THE ACCESS PATH THAT FINDS A PARTICULAR CHANGE NUMBER	[CHANGE_NO] —> CHANGE —> [PROG_ID] —> PERSONNEL ↓ [FORM_ NAME]
M1	PERSONNEL	FORM_NAME	NONE	—> PERSONNEL —> [FORM_NAME]
M2	PERSONNEL	FULL_NAME	PROGRAMMER FORM NAME	[FORM_NAME] —> PERSONNEL —> [FULL_NAME]
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_HR	PROJECT NAME	[PROJ_NAME] —> PROJECT ↓

5063-11/2-89[5-7]

Table 4-4. SEL Database Access Paths (9 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
P135,D94 (CONT'D)				↓ [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]
P3	PROJECT	ACTIVE _STATUS	PROJECT NAME	[PROJ_NAME] —>PROJECT ↓ [ACTIVE_STATUS]*CODED FIELD
P1, D1	PROJECT	PROJ_NAME	NONE	—> PROJECT ↓ [PROJ_NAME]
P2	PROJECT	PROJ_TYPE	PROJECT NAME	[PROJ_NAME] —>PROJECT ↓ [PROJ_TYPE]*CODED FIELD
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]
P23, 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]

5063-12/2-89[5-7]

Table 4-4. SEL Database Access Paths (10 of 18)

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 SUBMISSION 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 SUBMISSION 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 SUBMISSION 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 SUBMISSION 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 SUBMISSION 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 SUBMISSION DATE OF DESIRED SET OF ESTIMATES	[PROJ_NAME] —>PROJECT ↓ [PROJ_NO] [SUB_DATE] —>PROJ_EST ↓ [T_SYS]

5063-13/2-89[5-7]

Table 4-4. SEL Database Access Paths (11 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
D10, D91	PROJ_EST _PHASE	END_DATE	PROJECT NAME AND SUBMIS-SION DATE OF DESIRED SCHEDULE	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] [SUB_DATE] —> PROJ_EST_PHASE ↓ MAX [END_DATE]
D3, D4, D5, D6, D7, D8, D9, D84 TO D90	PROJ_EST _PHASE	START_DATE	PROJECT NAME, PHASE CODE, AND SUBMISSION DATE	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] [PHASE_CO] —> PROJ_EST_PHASE [SUB_DATE] ↓ [START_DATE] NOTE: PHASE_CO FOR D3, D84 = REQNT PHASE_CO FOR D4, D85 = DESGN PHASE_CO FOR D5, D86 = CODET PHASE_CO FOR D6, D87 = SYSTE PHASE_CO FOR D7, D88 = ACCTE PHASE_CO FOR D8, D89 = CLEAN PHASE_CO FOR D9, D90 = MAINT SUB_DATE FOR D3 TO D9 IS THE SUBMISSION DATE OF DESIRED SCHEDULE. SUB_DATE FOR D84 TO D90 IS THE SUBMISSION DATE OF FINAL STATISTICS.
P6, P7, P8, P9, P10, P11, P12, P125 TO P131	PROJ_EST _PHASE	START_DATE, END_DATE	PROJECT NAME, SUBMISSION DATE OF DESIRED SCHEDULE, AND PHASE CODE	[PROJ_NAME] —>PROJECT ↓ [PROJ_NO] [SUB_DATE] —> PROJ_EST_PHASE <—[PHASE_CO] ↓ [START_DATE], [END_DATE] NOTE: PHASE_CO FOR P6, P125 = REQNT PHASE_CO FOR P7, P126 = DESGN PHASE_CO FOR P8, P127 = CODET PHASE_CO FOR P9, P128 = SYSTE PHASE_CO FOR P10, P129 = ACCTE PHASE_CO FOR P11, P130 = CLEAN PHASE_CO FOR P12, P131 = MAINT
P5, P124, P13, 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 ↓

5063-14/2-99[5-7]

Table 4-4. SEL Database Access Paths (12 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
D20, D49, D113, D150 (CONT'D)				[PROJ_NO] ↓ [FORM_TYPE] —> PROJ_FORM ↓ [FORM_NO] NOTE: FORM_TYPE FOR D150 = SEF FORM_TYPE FOR D20 = PEF FORM_TYPE FOR D49 = SPF FORM_TYPE FOR D113 = PCSF
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	PROJ_MESS	MESSAGE	PROJECT NAME	[PROJ_NAME] —>PROJECT ↓ [PROJ_NO] PROJ_MESS ↓ [MESSAGE]
P45, D39	PROJ_PROD	RES_HR	PROJECT NAME, COMPUTER NAME, AND SUBMISSION DATE	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] [SUB_DATE] —> PROJ_PROD <— [RES_NAME] ↓ [RES_HR]

5063-15/2-89[5-7]

Table 4-4. SEL Database Access Paths (13 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
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 SUBMISSION DATE	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] [SUB_DATE] —> PROJ_PROD <— [RES_NAME] ↓ [RES_RUN]
P88 TO P107, P109 TO P123	PROJ_SEF	EVALUATE	PROJECT NAME AND MEASURE-MENT TYPE	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] [MEAS_TYPE] —> PROJ_SEF ↓ [EVALUATE] NOTE: MEAS_TYPE FOR P88, D14 IS 'PM01' MEAS_TYPE FOR P89, D115 IS 'PM02' MEAS_TYPE FOR P90, D116 IS 'PM03' MEAS_TYPE FOR P91, D117 IS 'PM04' MEAS_TYPE FOR P92, D118 IS 'PM05' MEAS_TYPE FOR P93, D119 IS 'PM06' MEAS_TYPE FOR P94, D120 IS 'ST07' MEAS_TYPE FOR P95, D121 IS 'ST08' MEAS_TYPE FOR P96, D122 IS 'ST09' MEAS_TYPE FOR P97, D123 IS 'ST10' MEAS_TYPE FOR P98, D124 IS 'TM11' MEAS_TYPE FOR P99, D125 IS 'TM12' MEAS_TYPE FOR P100, D126 IS 'TM13' MEAS_TYPE FOR P101, D127 IS 'TM14' MEAS_TYPE FOR P102, D128 IS 'TM15' MEAS_TYPE FOR P103, D129 IS 'PC16' MEAS_TYPE FOR P104, D130 IS 'PC17' MEAS_TYPE FOR P105, D131 IS 'PC18' MEAS_TYPE FOR P106, D132 IS 'PC19' MEAS_TYPE FOR P107, D133 IS 'PC20' MEAS_TYPE FOR P108, D134 IS 'PC21' MEAS_TYPE FOR P109, D135 IS 'PC22' MEAS_TYPE FOR P110, D136 IS 'PC23' MEAS_TYPE FOR P111, D137 IS 'PC24' MEAS_TYPE FOR P112, D138 IS 'EN25' MEAS_TYPE FOR P113, D139 IS 'EN26' MEAS_TYPE FOR P114, D140 IS 'EN27' MEAS_TYPE FOR P115, D141 IS 'EN28' MEAS_TYPE FOR P116, D142 IS 'EN29' MEAS_TYPE FOR P117, D143 IS 'EN30' MEAS_TYPE FOR P118, D144 IS 'PT31' MEAS_TYPE FOR P119, D145 IS 'PT32' MEAS_TYPE FOR P120, D146 IS 'PT33' MEAS_TYPE FOR P121, D147 IS 'PT34' MEAS_TYPE FOR P122, D148 IS 'PT35' MEAS_TYPE FOR P123, D149 IS 'PT36'

5063-35/2-89[5-7]

Table 4-4. SEL Database Access Paths (14 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
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 IS PC21
P133, D93	PROJ_STAT	SER_HR	PROJECT NAME	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [SER_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]
P132, D92	PROJ_STAT	TECH_MAN_HR	PROJECT NAME	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [TECH_MAN_HR]

5063-17/2-89[5-7]

Table 4-4. SEL Database Access Paths (15 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
P146, D105	PROJ_STAT	T_EXE_MOD	PROJECT NAME	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_EXE_MOD]
P150, D109	PROJ_STAT	T_EXE_STAT	PROJECT NAME	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_EXE_STAT]
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]
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]
P142, D101	PROJ_STAT	T_NEW_LINE	PROJECT NAME	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_NEW_LINE]

5063-18/2-89[5-7]

Table 4-4. SEL Database Access Paths (16 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
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]
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	PROJ_STAT	T_OLD_STAT	PROJECT NAME	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_OLD_STAT]
P137, D96	PROJ_STAT	T_SYS	PROJECT NAME	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] PROJ_STAT ↓ [T_SYS]
P150, D151	PROJ_SUB	SUB_DATE	PROJECT NAME AND SUBSYSTEM PREFIX	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] [SUB_PRE] —> PROJ_SUB ↓ [SUB_DATE]

5063G-(6)-42

Table 4-4. SEL Database Access Paths (17 of 18)

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
P47, D51,D152	PROJ_SUB	SUB_PRE	PROJECT NAME	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] PROJ_SUB ↓ [SUB_PRE]
P35, P36, P37, P38, D33 THROUGH D36	SPECIAL_ACT	ACT_HR	PROJECT NAME, PROGRAMMER NAME, AND WEEK ENDING DATE	[PROJ_NAME] —>PROJECT [PROJ_NO] [FORM_NAME] —>PERSONNEL ↓ [PROG_ID] —> EFF_PROJ <— [SUB_DATE] ↓ [P_ID] = [EFF_ID] [ACTIVITY] —> SPECIAL_ACT ↓ [ACT_HR] WHERE SP_ACTIVITY FOR P35, D33 = REWORK SP_ACTIVITY FOR P36, D34 = ENHANCE SP_ACTIVITY FOR P37, D35 = DOCUMENT SP_ACTIVITY FOR P38, D36 = REUSE
P52	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]
P51, D53	SUB_COM	COM_NAME	PROJECT NAME AND SUB-SYSTEM PREFIX	[PROJ_NAME] —>PROJECT ↓ [PROJ_NO] [SUB_PRE] —>PROJ_SUB ↓ [SUBSY_ID] SUB_COM ↓ [COM_NAME]
P49, D154	SUBSYSTEM	FUNCTION	PROJECT NAME AND SUBSYSTEM PREFIX	[PROJ_NAME] —>PROJECT ↓ [PROJ_NO] [SUB PRE] —>PROJ_SUB ↓

5063G-(6)-43

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

REF. ID	TARGET TABLE	TARGET COLUMN	ACCESS INFORMATION	ACCESS PATH
P49, D154 (CONT'D)				↓ [SUBSY_ID] SUBSYSTEM ↓ [FUNCTION]* CODED FIELD
P48, D153	SUBSYSTEM	NAME	PROJECT NAME AND SUB-SYSTEM PREFIX	[PROJ_NAME] —> PROJECT ↓ [PROJ_NO] [SUB_PRE] —> PROJ_SUB ↓ [SUBSY_ID] SUBSYSTEM ↓ [NAME]
P84, D 62	V_PROJ_COM	COM_NAME	PROJECT NAME	CHANGE_COM ↓ [COM_NO] [PROJ_NAME] —> V_PROJ_COM ↓ [COM_NAME]

5063-21/2-89[5-7]

SECTION 5 - ACCESSING THE SEL DATABASE

The database table definitions and relationships 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 gets access to the SEL database. Section 5.2 provides an introduction to the Database Access Manager for the SEL (DAMSEL) software system: a menu-driven user interface that allows the user 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 the SQL language provided by the ORACLE DBMS. This introduction covers the basics of how to formulate an SQL query and provides several illustrative examples.

5.1 DATABASE ACCESS REQUIREMENTS

To access the SEL database, a user must first have a user ID on the STL VAX 11/780. Users can register for this account by contacting STL systems personnel. Second, the user must have an ORACLE user ID on the VAX. This may be obtained from STL ORACLE systems personnel. Third, the user must be enrolled as a database user. This may be accomplished by contacting the CSC SEL DBA and supplying an ORACLE user ID, password, and SEL database user class. User classes are defined to give different types of users different 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 under the database operational software. 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 has been enrolled in the SEL database environment and logs onto the STL VAX, the following command procedure must be executed 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 to access the database, it is recommended that it be included in the user's LOGIN.COM file to be executed automatically upon logging onto the VAX. Then, after logging on, the user may execute the DAMSEL system by simply typing

```
$ DAMSEL
```

5.2 DAMSEL SYSTEM

The DAMSEL system is the primary facility that provides a convenient way to access the SEL data for all classes of users. This is a menu-driven user interface with five major options at the top level:

- Forms function option--Users may 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 from the report menus and submit the sequence to be executed. They may also save one or more frequently used sequences of reports for future execution. Reports are submitted as batch jobs, and the results may be printed or routed to files for terminal display and 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.

- 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 commands to SEL database support personnel to back up and restore the database and to generate distribution tapes.

In the menu system, users, depending on their user class, may access 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 support function backup and distribution tape generation 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 backup and distribution tape generation 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 general support function screens, and all the DBA function screens.

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

- Log-on to the VAX under his/her VAX account.

- At the '$' prompt, type DAMSEL.

- Enter his/her ORACLE user name and password on the first screen in the menu system.

- Select menu options.
- Terminate the menu system session via the <Exit/Cancel> key.

Reference 3 presents a more detailed discussion on using the operational 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 the operational software 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. The reader is referred 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 STL 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, 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 an SQL command. Upon entering an SQL command, terminated with a semicolon (;), and pressing "return," SQL processes the command, displays the result, and returns to the SQL> prompt.

While in an 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 an SQL*Plus session, the user types

```
SQL> EXIT
```

to disconnect from ORACLE and return to the system prompt.

5.3.2 BASIC SELECT STATEMENT

The SQL statement for retrieving database 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 to the terminal 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	SIM	ACT_DEV
PROJ_102	102	AGSS	ACT_DEV
PROJ_103	103	SIM	ACT_DEV
PROJ_104	104	SIM	ACT_DEV
PROJ_105	105	AGSS	ACT_DEV
PROJ_106	106	SIM	ACT_DEV
PROJ_71	71	SIM	INACTIVE
PROJ_110	110	AGSS	ACT_DEV
PROJ_108	108	SIM	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. It should be noted that the command typed in 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 = 'SIM';
```

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

This query selects only those records in which the PROJ_TYPE column has a value of 'SIM'. 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           SIM             ACT_DEV
PROJ_101          101           SIM             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 = 'SIM'
  3  AND   PROJ_NO > 104;

PROJ_NAME       PROJ_NO       PROJ_TYPE       ACTIVE_STATUS

PROJ_106          106           SIM             ACT_DEV
PROJ_108          108           SIM             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 the count of all 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 the 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, the concept of a subquery is introduced. 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
  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. It should be noted 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 to the terminal. 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 is done, any commands entered and the associated results displayed are spooled to this file. Spooling can be turned off, with the following command:

```
SQL> SPOOL OFF;
```

Another useful feature is to be able 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 command may now be executed or listed with / or L as described above.

This section has presented enough of an introduction to ad hoc database queries to enable the user to access any particular item of software engineering data in which he/she is interested. It has not, however, covered all of the features present 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 columns and defining display formats, parameterizing queries, and 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 are referred to Reference 4.

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.

Field Where Used	Value (Description)	Code
ACTIVE_STATUS	Data collection is active; project is in development	ACT_DEV
ACTIVE_STATUS	Data collection is active; project is in maintenance	ACT_MAINT
ACTIVE_STATUS	Data for the project are incomplete; no plan to validate data	DISCONT
ACTIVE_STATUS	The project has been completed and no more data are being collected	INACTIVE
ACTIVITY	Pre design	PREDES
ACTIVITY	Create design	CREDES
ACTIVITY	Read/review code	RDREVCOD
ACTIVITY	Write code	WRCODE
ACTIVITY	Read/review design	RDREVDES
ACTIVITY	Test code units	TSTCODUN
ACTIVITY	Debugging	DEBUG
ACTIVITY	Integration test	INTTEST
ACTIVITY	Acceptance test	ACCTEST
ACTIVITY	Other	OTHER
ACTIVITY	Support	SUPPORT
ADA_FEATURE	Data typing	DATATYPE
ADA_FEATURE	Subprograms	SUBPROG
ADA_FEATURE	Exceptions	EXCEPT
ADA_FEATURE	Generics	GEN
ADA_FEATURE	Program structure and packaging	PACK
ADA_FEATURE	Tasking	TASK
ADA_FEATURE	System dependent features	SYSDEPF
ADA_FEATURE	Other	OTHER
CH_TYPE	Error correction	ERRCO
CH_TYPE	Planned enhancement	PLANE

Field Where Used	Value (Description)	Code
CH_TYPE	Implementation of requirements change	IMPRE
CH_TYPE	Improvement of clarity, maintainability, or documentation	IMPCM
CH_TYPE	Improvement of user services	IMPUS
CH_TYPE	Insertion/deletion of debug code	IN/DE
CH_TYPE	Optimization of time/space/accuracy	OPTSA
CH_TYPE	Adaptation to environment change	ADENC
CH_TYPE	Other change type	OTHCH
COM_TYPE	Include file	INCL
COM_TYPE	Job control language	JCL
COM_TYPE	Assembly language component	ALC
COM_TYPE	FORTRAN source code	FORTRAN
COM_TYPE	Pascal source code	PASCAL
COM_TYPE	NAMELIST or parameter list	NAMELT
COM_TYPE	Display identification	DISPLAY
COM_TYPE	Menu definition or help file	MENDEF
COM_TYPE	Reference data file	REFDATA
COM_TYPE	BLOCK DATA component	BLOCKDA
COM_TYPE	Ada subprogram specification	ADASUBS
COM_TYPE	Ada subprogram body	ADASUBB
COM_TYPE	Ada package specification	ADAPACKS
COM_TYPE	Ada package body	ADAPACKB
COM_TYPE	Ada task specification	ADATASKS
COM_TYPE	Ada task body	ADATASKB
COM_TYPE	Ada generic specification	ADAGENS
COM_TYPE	Ada generic body	ADAGENB
COM_TYPE	Other type of component	OTHER
COM_TYPE	Ada source code (type unspecified)	ADAUNSPEC
EFF_COM_CH	1 hour or less	1HR
EFF_COM_CH	1 hour to 1 day	1DAY

Field Where Used	Value (Description)	Code
EFF_COM_CH	1 day to 3 days	3DAY
EFF_COM_CH	More than 3 days	NDAY
EFF_ISO_CH	1 hour or less	1HR
EFF_ISO_CH	1 hour to 1 day	1DAY
EFF_ISO_CH	1 day to 3 days	3DAY
EFF_ISO_CH	More than 3 days	NDAY
ERR_ACAUSE	Misunderstood interaction of features	INTERACT
ERR_ACAUSE	Features applied incorrectly	INCOF
ERR_ACAUSE	Misunderstood features	FEATUREM
ERR_ACAUSE	Confused features	FEATUREC
ERR_ARES	Class notes	NOTE
ERR_ARES	Ada reference manual	REFMAN
ERR_ARES	Own project team member	TEAM
ERR_ARES	Own memory	MEMORY
ERR_ARES	Someone not on project team	NTEAM
ERR_ARES	Other	OTHER
ERR_CLASS	Initialization	INIT
ERR_CLASS	Logic/control structure	LOGIC
ERR_CLASS	Interface (internal)	INTERI
ERR_CLASS	Interface (external)	INTERE
ERR_CLASS	Data value or structure	DATAVAL
ERR_CLASS	Computational	COMPUTE
ERR_SOURCE	Requirements	REQMT
ERR_SOURCE	Functional specifications	FUNSPEC
ERR_SOURCE	Design	DESIGN
ERR_SOURCE	Code	CODE
ERR_SOURCE	Previous change	PRECH
ERR_TOOLS	Compiler	COMPI
ERR_TOOLS	Symbolic debugger	SYMDEB
ERR_TOOLS	Language sensitive editor	LSE
ERR_TOOLS	CMS	CMS
ERR_TOOLS	Source code analyzer	SCA

Field Where Used	Value (Description)	Code
ERR_TOOLS	Performance and coverage analyzer	PCA
ERR_TOOLS	DEC Test Manager	DECTM
ERR_TOOLS	Other	OTHER
FUNCTION	User interface	USERINT
FUNCTION	Data processing/data conversion	DPDC
FUNCTION	Real-time control	REALTIME
FUNCTION	Mathematical/computational	MATHCOMP
FUNCTION	Graphics and special device support	GRAPH
FUNCTION	Control processing/executive	CPEXEC
FUNCTION	System services	SYSSERV
MEAS_TYPE	Problem difficulty	PM01
MEAS_TYPE	Tightness of schedule constraints	PM02
MEAS_TYPE	Requirements stability	PM03
MEAS_TYPE	Quality of specification documents	PM04
MEAS_TYPE	Requirements for documentation	PM05
MEAS_TYPE	Rigor of formal reviews	PM06
MEAS_TYPE	Ability of development team	ST07
MEAS_TYPE	Development team experience with application	ST08
MEAS_TYPE	Development team experience with environment	ST09
MEAS_TYPE	Stability of development team composition	ST10
MEAS_TYPE	Project management performance	TM11
MEAS_TYPE	Project management experience with application	TM12
MEAS_TYPE	Stability of project management team	TM13
MEAS_TYPE	Project planning discipline	TM14
MEAS_TYPE	Degree project plans followed	TM15
MEAS_TYPE	Modern programming practices	PC16
MEAS_TYPE	Disciplined change/question tracking	PC17

Field Where Used	Value (Description)	Code
MEAS_TYPE	Use of disciplined requirements analysis methodology	PC18
MEAS_TYPE	Use of disciplined design methodology	PC19
MEAS_TYPE	Use of disciplined testing methodology	PC20
MEAS_TYPE	Use of tools	PC21
MEAS_TYPE	Use of test plans	PC22
MEAS_TYPE	Use of quality assurance procedures	PC23
MEAS_TYPE	Use of configuration management procedures	PC24
MEAS_TYPE	Degree of access to development system	EN25
MEAS_TYPE	Programmers per terminal	EN26
MEAS_TYPE	Development machine resource constraints	EN27
MEAS_TYPE	System response time	EN28
MEAS_TYPE	System hardware and support software stability	EN29
MEAS_TYPE	Software tool effectiveness	EN30
MEAS_TYPE	Delivered software supports requirements	PT31
MEAS_TYPE	Quality of delivered software	PT32
MEAS_TYPE	Quality of design present in delivered software	PT33
MEAS_TYPE	Quality/completeness of software documentation	PT34
MEAS_TYPE	Timely software delivery	PT35
MEAS_TYPE	Smoothness of acceptance testing	PT36
MESS_TYPE	Computer accounts to monitor	COMPACC
MESS_TYPE	Names of controlled libraries	CONLIB
MESS_TYPE	CSC contact	CSCP
MESS_TYPE	Current phase	CURPH
MESS_TYPE	Development machine	DEVMA

Field Where Used	Value (Description)	Code
MESS_TYPE	Growth history tool used	GHTOOL
MESS_TYPE	GSFC contact	GSFCP
MESS_TYPE	SEL forms required	SELF
MESS_TYPE	Task numbers and corresponding years	TASKNO
MESS_TYPE	Text comment 1	TEXT1
MESS_TYPE	Text comment 2	TEXT2
MESS_TYPE	Text comment 3	TEXT3
MESS_TYPE	Text comment 4	TEXT4
MESS_TYPE	Text comment 5	TEXT5
MESS_TYPE	Text comment 6	TEXT6
MESS_TYPE	Text comment 7	TEXT7
MESS_TYPE	Text comment 8	TEXT8
MESS_TYPE	Text comment 9	TEXT9
MESS_TYPE	Text comment 10	TEXT10
ORI_TYPE	New	NEW
ORI_TYPE	Extensively modified	EXTMO
ORI_TYPE	Slightly modified	SLMOD
ORI_TYPE	Old (unchanged)	OLDUC
PHASE_CO	Requirements definition	REQNT
PHASE_CO	Design	DESGN
PHASE_CO	Code and test (implementation)	CODET
PHASE_CO	System test	SYSTE
PHASE_CO	Acceptance test	ACCTE
PHASE_CO	Cleanup	CLEAN
PHASE_CO	Maintenance	MAINT
PROJ_TYPE	Attitude oriented	ATTITUDE
PROJ_TYPE	Other	OTHER
PROJ_TYPE	Attitude ground support system	AGSS
PROJ_TYPE	Simulator	SIM
PROJ_TYPE	Orbit oriented	ORBIT
PROJ_TYPE	Scientific oriented	SCIENTIFIC

Field Where Used	Value (Description)	Code
PROJ_TYPE	Database	DATABASE
PROJ_TYPE	Real time processing	REALTIME
PROJ_TYPE	Software tool	TOOL
PURPOSE	I/O processing	IOPRO
PURPOSE	Algorithmic/computational	ALCOMP
PURPOSE	Data transfer	DATRA
PURPOSE	Logic/decision	LODEC
PURPOSE	Control module	CNTRMOD
PURPOSE	Interface to operating system	INTOP
PURPOSE	Ada process abstraction	ADAPR
PURPOSE	Ada data abstraction	ADADA
QA_STATUS	Hand-checked: errors found	HCERROR
QA_STATUS	Hand-checked: correct	HCCORRECT
SECOND_L	Compiler	COMPI
SECOND_L	Linker	LINK
SECOND_L	Editor	EDIT
SECOND L	Graphics display builder	GRADIS
SECOND_L	Requirements language processor	REPLP
SECOND_L	Structured analysis tool	STRANT
SECOND_L	PDL processor	PDLPR
SECOND_L	ISPF	ISPF
SECOND_L	Source Code Analyzer Program	SAP
SECOND_L	Configuration Analysis Tool	CAT
SECOND_L	PANVALET	PANVAL
SECOND_L	Test coverage tool	TESTCO
SECOND_L	Interface checker (e.g., RXVP80, - ANALYZ)	INTERF
SECOND_L	Language sensitive editor	LSE
SECOND_L	Symbolic debugger	SYMDEB
SECOND_L	Configuration management tool (e.g., CMS, MMS)	CMTOOL
SECOND_L	Other tools	OTHER
SECOND_L	Software development environment	SDE

Field Where Used	Value (Description)	Code
SP_ACTIVITY	Rework	REWORK
SP_ACTIVITY	Enhance/refine/optimize	ENHANCE
SP_ACTIVITY	Document	DOCUMENT
SP_ACTIVITY	Reuse	REUSE
STATUS	Unchecked	UNCHK
STATUS	Hand-checked: correct	HCCORRECT
STATUS	Verified by application	VERAP
STATUS	Hand-checked: errors found	HCERROR

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 DBMS environment. In each example, the data desired from the database are 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 realization 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 PRF forms 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. Retrieves all manpower hours charged during the design phase.

      First, find the design phase start and end dates.

```
SQL> SELECT START_DATE,END_DATE
     FROM   PROJ_EST_PHASE,PROJECT
     WHERE  SUB_DATE =
      (SELECT MAX(SUB_DATE)
      FROM PROJ_EST_PHASE
      WHERE PROJ_NO = PROJECT.PROJ_NO)
     AND PHASE_CO = 'DESGN'
     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 - GLOSSARY OF TERMS AND ABBREVIATIONS

TERMS

Clause	A portion of an 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.
Column	A particular class of data items within a table. Each column has a single value in each row of a table.
Command	An instruction to the SQL*Plus interpreter.
Field	Synonymous with column.
Group Function	An 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 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.
Primary Key	One or more columns whose values uniquely identify each 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.
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.
View	A "virtual table" that consists of one or more columns from underlying database tables. Views do not actually store data.

ABBREVIATIONS

AGSS	Attitude Ground Support System
CDR	critical design review
COF	Component Origination Form
CPU	central processing unit
CRF	Change Report Form
DBA	database administrator
DBMS	database management system
DDL	data definition language
GSFC	Goddard Space Flight Center
ID	identification
NASA	National Aeronautics and Space Administration
PCSF	Project Completion Statistics Form
PDL	program design language
PDR	preliminary design review
PEF	Project Estimates Form
PRF	Personnel Resource Form
SEF	Subjective Evaluation Form
SEL	Software Engineering Laboratory
SIF	Subsystem Information Form
SPF	Services/Products Form
SQL	structured query language
STL	Systems Technology Laboratory

APPENDIX D - SEL DATA COLLECTION FORMS

This appendix contains all the SEL data collection forms. These forms are completed by programmers and managers of SEL-monitored projects, with the exception of one form, the Service/Products form, that is completed by SEL personnel.

PROJECT ESTIMATES FORM

Project Name: D1

Form Date: D2

Phase Dates (Saturdays)	
Phase	Start Date
Requirements	D3
Design	D4
Code & Test	D5
System Test	D6
Acceptance Test	D7
Cleanup	D8
Maintenance	D9
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:

6037-21

JULY 1987

Figure D-1. Project Estimates Form

Personnel Resources Form

Name: D21

Project: D1 Friday Date: 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:

6037-23

JULY 1987

Figure D-2. Personnel Resources Form

SERVICES/PRODUCTS FORM

Project: D1

Friday Date: D22

Computer	CPU-hours	No. of runs
D38	D39	D40

Modules	D41
Changes	D42
Lines of Code	D43

Service	Hours
Tech Pubs	D44
Secretary	D45
Librarians	D46
Other	D47
Proj. Mgmt.	D48

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

5063-25

JULY 1987

Figure D-3. Services/Products Form

COMPONENT ORIGINATION FORM

Project Name: D1 Programmer Name: D50

Subsystem Prefix: D51 Form Date: D52

Component Name: D53

Date entered into controlled library: D54

Location of Developer's Source File

Library or directory: _____

Member name: _____

Relative Difficulty of Developing Component

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

Easy		Medium		Hard	D55
1	2	3	4	5	

Origin

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 D56
_____ Extensively modified (more than 25% of statements changed)
_____ Slightly modified
_____ Old (unchanged)

For Librarian's Use Only
Number: D59
Date: _____
Entered by: _____
Checked by: _____

If not new, what project or library is it from? _____

Type of Component (Check one only) D57

_____ 'INCLUDE' file (e.g., COMMON)
_____ JCL (or other control)
_____ ALC (assembler code)
_____ FORTRAN executable source
_____ Pascal source
_____ NAMELIST or parameter list
_____ Display identification (GESS)
_____ Menu definition or help
_____ Reference data files
_____ BLOCK DATA file
_____ Ada subprogram specification
_____ Ada subprogram body
_____ Ada package specification
_____ Ada package body
_____ Ada task specification
_____ Ada task body
_____ 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
_____ Ada process abstraction
_____ Ada data abstraction

6063-27

JULY 1987

Figure D-4. Component Origination Form

CHANGE REPORT FORM

Project Name: D1 Current Date: D60

Programmer Name: D61 Approved by:

Section A – Identification

Describe the change: (What, why, how)

Effect: What components (or documents) are changed? (Include version) D62

Effort: What additional components (or documents) 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 project is in Ada (If so, complete questions on reverse side)*

	1 hr/less	1 hr/1 dy	1 dy/3 dys	>3 dys
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)

D67

- ☐ 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 (Explain on back)

Effects of Change

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

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)*

D72

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

For Librarian's Use Only

Number: D82

Date:

Entered by:

Checked by:

5063G(6)-33

JULY 1987

Figure D-5. Change Report Form (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:

   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 the evaluation of the change and the use of Ada

JULY 1988

Figure D-5. Change Report Form (2 of 2)

SUBSYSTEM INFORMATION FORM

Project Name: D1

Date: D151

Subsystem Prefix	Subsystem Name	Subsystem Function
D151	D153	D154

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.

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

5063G(6)-34

JULY 1987

Figure D-6. Subsystem Information Form

PROJECT COMPLETION STATISTICS FORM

Project Name: D1

Form Date: D83

Phase Dates (Saturdays)	
Phase	Start Date
Requirements	D84
Design	D85
Code & Test	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	Modified	D102
Pages of documentation	D99	Old	D103
		Comments	D104
Executable Modules		Executable Statements	
Total	D105	Total	D109
New	D106	New	D110
Modified	D107	Modified	D111
Old	D108	Old	D112

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:

6037-34

JULY 1987

Figure D-7. Project Completion Statistics Form

Name P24

Project Name P1

Submission Date P13

SUBJECTIVE EVALUATION FORM

Purpose: To obtain subjective assessments on recently completed software development projects.

Completed by: Personnel participating in management of the project, within one month of project completion.

I. PROBLEM CHARACTERISTICS

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

1	2	3	4	5
Easy		Average	D114	Difficult

2. How tight were the schedule constraints on the project?

1	2	3	4	5
Loose		Average	D115	Tight

3. How stable were the requirements over the development period?

1	2	3	4	5
Low		Average	D116	High

FOR LIBRARIAN'S USE ONLY

Number: D150 Entered by: _____

Date: _____ Checked by: _____

Figure D-8. Subjective Evaluation Form (1 of 8)

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

1	2	3	4	5
Low		Average	D117	High

5. How extensive were the documentation requirements?

1	2	3	4	5
Low		Average	D118	High

6. How rigorous were the formal review requirements?

1	2	3	4	5
Low		Average	D119	High

II. PERSONNEL CHARACTERISTICS: TECHNICAL STAFF

7. Assess the overall quality and ability of the development team.

1	2	3	4	5
Low		Average	D120	High

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

1	2	3	4	5
Low		Average	D121	High

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

1	2	3	4	5
Low		Average	D122	High

JULY 1987

Figure D-8. Subjective Evaluation Form (2 of 8)

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

1	2	3	4	5
Low		Average	D123	High

III. PERSONNEL CHARACTERISTICS: TECHNICAL MANAGEMENT

11. Assess the overall performance of project management.

1	2	3	4	5
Low		Average	D124	High

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

1	2	3	4	5
Low		Average	D125	High

13. How stable was the project management over the duration of the project?

1	2	3	4	5
Low		Average	D126	High

14. What degree of disciplined project planning was used?

1	2	3	4	5
Low		Average	D127	High

15. To what degree were project plans followed?

1	2	3	4	5
Low		Average	D128	High

JULY 1987

Figure D-8. Subjective Evaluation Form (3 of 8)

IV. PROCESS CHARACTERISTICS

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

1	2	3	4	5
Low		Average	D129	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?

1	2	3	4	5
Low		Average	D130	High

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

1	2	3	4	5
Low		Average	D131	High

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

1	2	3	4	5
Low		Average	D132	High

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

1	2	3	4	5
Low		Average	D133	High

JULY 1987

Figure D-8. Subjective Evaluation Form (4 of 8)

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.

- ☐ Compiler
- ☐ Linker
- ☐ Editor **D134**
- ☐ 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?

1	2	3	4	5
Low		Average	**D135**	High

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

1	2	3	4	5
Low		Average	**D136**	High

Figure D-8. Subjective Evaluation Form (5 of 8)

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

1	2	3	4	5
Low		Average	D137	High

V. ENVIRONMENT CHARACTERISTICS

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

1	2	3	4	5
Low		Average	D138	High

26. What was the ratio of programmers to terminals?

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

1	2	3	4	5
Low		Average	D140	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?

1	2	3	4	5
Poor		Average	D141	Very Good

JULY 1987

Figure D-8. Subjective Evaluation Form (6 of 8)

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

1	2	3	4	5
Low		Average	D142	High

30. Assess the effectiveness of the software tools.

1	2	3	4	5
Low		Average	D143	High

VI. PRODUCT CHARACTERISTICS

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

1	2	3	4	5
Low		Average	D144	High

32. Assess the quality of the delivered software product.

1	2	3	4	5
Low		Average	D145	High

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

1	2	3	4	5
Low		Average	D146	High

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

1	2	3	4	5
Low		Average	D147	High

JULY 1987

Figure D-8. Subjective Evaluation Form (7 of 8)

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

1	2	3	4	5
Low		Average	D148	High

36. Assess the smoothness or relative ease of acceptance testing.

1	2	3	4	5
Low		Average	D149	High

JULY 1987

Figure D-8. Subjective Evaluation Form (8 of 8)

APPENDIX E - 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.

In the 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 data type.

The primary key component(s) is identified by the keyword KEY, and a unique index will be created for every primary key in the database. The keyword UNIQUE identifies the fields that are not part of the primary key but whose values are unique within a relation. The keyword INDEX identifies fields to be indexed in addition to the primary key field(s). CLUSTER identifies relations that are physically stored together.

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" (V) 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 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)
  CLUSTER (PROJ_SUB)

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_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_SUB
    (FIELD PROJ_NO numeric(3)
     FIELD SUB_PRE char(5)
     FIELD SUBSY_ID numeric(5))
  KEY (PROJ_NO, SUB_PRE)
  UNIQUE (SUBSY_ID)
  INDEX (SUBSY_ID)
  CLUSTER (PROJECT)
  CONSTRAINT
      RANGE PROJECT P
      RANGE PROJ_SUB S
              ∀S  ∃P  (P.PROJ_NO = S.PROJ_NO)
```

```
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_NO, 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'  VPF.FORM_TYPE
                        = 'SPF'  VPF.FORM_TYPE = 'PCSF'V
                        PF.FORM_TYPE = 'SEF')

RELATION PROJ_STAT
    (FIELD PROJ_NO numeric(3)
     FIELD SUB_DATE date
     FIELD T_SYS numeric(4)
     FIELD T_COM numeric(4)
     FIELD T_EXE_MOD numeric(4)
     FIELD T_NEW_MOD numeric(4)
     FIELD T_MOD_MOD numeric(4)
     FIELD T_EXE_STAT numeric(6)
     FIELD T_NEW_STAT numeric(6)
     FIELD T_CH numeric(6)
     FIELD T_LINE numeric(7)
     FIELD T_DOC numeric(6)
     FIELD T_NEW_LINE numeric(6)
     FIELD T_MOD_LINE numeric(6)
     FIELD T_MOD_STAT numeric(6)
     FIELD T_OLD_LINE numeric(6)
     FIELD T_OLD_STAT numeric(6)
     FIELD T_OLD_MOD numeric(4)
     FIELD PRO_HR numeric(10,2)
     FIELD TECH_MAN_HR numeric(10,2)
     FIELD SER_HR numeric(10,2)
     FIELD T_COMMENT numeric(6))
  KEY (PROJ_NO, SUB_DATE)
  CONSTRAINT
      RANGE PROJECT P
      RANGE PROJ_EST PES
              ∀PES  ∃P (P.PROJ_NO = PES.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
      RANGE VAL_CPU_PURPOSE VCP
              ∀ PESC  ∃ P (P.PROJ_NO = PESC.PROJ_NO)
              ∀ PESC  ∃ CPU (CPU.CPU_NAME = PESC.CPU_NAME)

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 day)
              ∀ PESP  ∃ PESP (PESP.END_DATE = a valid
                              Saturday day)

RELATION PROJ_MESS
    (FIELD PROJ_NO numeric(3)
     FIELD MESS_TYPE char(10)
     FIELD MESSAGE char(65)
     FIELD DATE_ENTRY date)
  KEY (PROJ_NO, MESS_TYPE)
  CONSTRAINT
      RANGE PROJECT P
      RANGE PROJ_MESS PE
      RANGE VAL_MESS_TYPE VMET
              ∀ PE  ∃ P (P.PROJ_NO = PE.PROJ_NO)
              ∀ PE  ∃ VMET (VMET.CODE = PE.MESS_TYPE)

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 VALIDATION
    (FIELD F_NAME char(20)
     FIELD CODE char(10)
     FIELD VALUE char(75))
  KEY (F_NAME, CODE)

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 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 COM_PURPOSE
    (FIELD COM_NO numeric(7)
     FIELD PURPOSE char(10))
  KEY (COM_NO, PURPOSE)
  CONSTRAINT
      RANGE SUB_COM 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_STAT
    (FIELD COM_NO numeric(7)
     FIELD C_EXE_S numeric(6)
     FIELD C_LINE numeric(6)
     FIELD C_C_LINE numeric(6))
  KEY (COM_NO)
  CONSTRAINT
      RANGE SUB_COM C
      RANGE COM_STAT CS
              ∀CS  ∃C (C.COM_NO = CS.COM_NO)

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 (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 CHANGE_COM
    (FIELD CHANGE_NO char(6)
     FIELD COM_NO numeric(7))
  KEY (CHANGE_NO, 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 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
      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 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_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 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 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 COMPUTER
    (FIELD CPU_NAME char(10)
     FIELD C_FULL_NAME char(20))
     KEY (CPU_NAME)

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 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)
  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'∨
                        EFF.FORM_TYPE = 'PRF')

RELATION EFF_SUPER
   (FIELD P_ID numeric(10)
    FIELD PER_SUPER numeric(6,2))
  KEY (P_ID)
  CONSTRAINT
      RANGE EFF_PROJ EP
      RANGE EFF_SUPER ESU
            ∀ESU  ∃EP (EP.P_ID = ESU.P_ID)

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  ∃ES  EP (ES.PS_ID = EA.EFF_ID ∨ EP.P_ID
                            = EA.EFF_ID)

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))
  KEY (SCRIPT_NO,SAT_DAY)
  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_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))
  KEY (SCRIPT_NO,SAT_DAY)
  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 TEMP_ACTIVITY
    (FIELD SAT_DAY date
     FIELD ACTIVITY char(8)
     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))
  KEY (SCRIPT_NO,SAT_DAY)
  CONSTRAINT
      RANGE TEMP_ACTIVITY TEMP
      RANGE GENERATE_SAT_DAY GSAT
              ∀TEMP  ∃GSAT  (GSAT.SCRIPT_NO = TEMP.SCRIPT_NO
                             ∧GSAT.SAT_DAY = TEMP.SAT_DAY)

RELATION TEMP_FORMCT
    (FIELD SUB_DAY date
     FIELD PROJ_NO numeric(3)
     FIELD PROG_ID numeric(5)
     FIELD FORM_TYPE char(6)
     FIELD SCRIPT_NO numeric(10))
  KEY (SCRIPT_NO,SAT_DAY)
  CONSTRAINT
      RANGE TEMP_FORMCT TEMP
      RANGE GENERATE_SAT_DAY GSAT
              ∀TEMP  ∃GSAT  (GSAT.SCRIPT_NO = TEMP.SCRIPT_NO
                             ∧GSAT.SAT_DAY = TEMP.SAT_DAY)
```

```
RELATION REP_CODES
    (FIELD CODE char(10)
     FIELD VALUE char(30)
     FIELD FUNCTION char(15))
  KEY (CODE)

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 DUMMY
    (FIELD HIDDEN char(1))

RELATION GENERATE_SAT_DAY
    (FIELD SAT_DAY date
     FIELD SCRIPT_NO numeric(10))
  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 PERM_SCRIPT
    (FIELD ORA_USER char(20)
     FIELD OUT_FILE char(20)
     FIELD OUT_ROUTING char(20)
     FIELD SCRIPT_NAME char(20)
     FIELD SCRIPT_NO numeric(10))
  KEY (ORA_USER, SCRIPT_NAME)
  UNIQUE SCRIPT_NO

  CONSTRAINT
      RANGE USER_CLASS U
      RANGE PERM_SCRIPT P
               ∀P  ∃U  (U.ORA_USER = P.ORA_USER)
               ∀P  ∃P  ((P.OUT_ROUTING = 'P')
                        ∧(P.OUT_FILE != null∧
                          P.OUT_ROUTING = 'F'))
```

```
RELATION REP_CONDITIONS
    (FIELD END_DATE date
     FIELD LINES_OF_CODE numeric(5)
     FIELD NUM_COM numeric(5)
     FIELD PROJ_TYPE char(10)
     FIELD REPORT_SEQ numeric(3)
     FIELD SCRIPT_NO numeric(10)
     FIELD START_DATE date)
  KEY (SCRIPT_NO, REPORT_SEQ)
  CONSTRAINT
      RANGE SCRIPT_REPORT S
      RANGE REP_CONDITIONS REP
              ∀REP  ∃S  (S.SCRIPT_NO = REP.SCRIPT_NO
                         S.REPORT_TYPE_SELECTION =
                         'SCONDITION'
                          ∧S.REPORT_SEQ = REP.REPORT_SEQ)

RELATION SCRIPT_PROJECTS
    (FIELD PROJ_NAME char(8)
     FIELD REPORT_SEQ numeric(3)
     FIELD SCRIPT_NO numeric(10))
  KEY (SCRIPT_NO,PROJ_NAME,REPORT_SEQ)
  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 REPORT_CODE char(10)
     FIELD REPORT_SEQ numeric(3)
     FIELD REPORT_TYPE char(20)
     FIELD REPORT_TYPE_SELECTION char(10)
     FIELD SCRIPT_NO numeric(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'
                                  ∨S.REPORT_TYPE_SELECTION =
                                    'ACTIVE'
                                  ∨S.REPORT_TYPE_SELECTION = 'ALL'
                                  ∨S.REPORT_TYPE_SELECTION = 'LIST'
                                  ∨S.REPORT_TYPE_SELECTION =
                                    'SCONDITION'
                                  ∧S.REPORT_TYPE = 'M') ∨
                                (S.REPORT_TYPE_SELECTION = null
                                  ∨S.REPORT_TYPE = 'O') ∨
                                (S.REPORT_TYPE_SELECTION =
                                  PROJ.PROJ_NAME ∧S.REPORT_TYPE =
                                  'S'))

RELATION SEQNO
    (FIELD FIELD_NAME char(30)
     FIELD MAXSEQNO numeric(10)
     FIELD TABLE_NAME char(30))
  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 ACT_HR numeric(10,2)
     FIELD EFF_ID numeric(10)
     FIELD SP_ACTIVITY char(10))
  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.EFF_ID = SA.EFF_ID
                              ∨ES.EFF_ID = SA.EFF_ID)
              ∀SA  ∃VAL  (VAL.SP_ACTIVITY = SA.SP_ACTIVITY)

RELATION TABLE_PRIVILEGE
    (FIELD ALTER_PRIV char(1)
     FIELD DELETE_PRIV char(1)
     FIELD INDEX_PRIV char(1)
     FIELD INSERT_PRIV char(1)
     FIELD SELECT_PRIV char(1)
     FIELD TABLE_NAME char(40)
     FIELD UPDATE_PRIV char(1)
     FIELD USER_CLASS char(20))
```

```
  KEY (TABLE_NAME,USER_CLASS)
  CONSTRAINT
      RANGE TABLE_PRIVILEGE T
      RANGE USER_CLASS U
              ∀T  ∃U (U.USER_CLASS = T.USER_CLASS)
              ∀T  ∃T (T.TABLE_NAME = a valid relation in the
                      database)
RELATION TEMP_SCRIPT
    (FIELD DELETE_STATUS char(10)
     FIELD ORA_USER char(20)
     FIELD OUT_FILE char(20)
     FIELD OUT_ROUTING char(20)
     FIELD PROCESS_ID char(20)
     FIELD RUN_STATUS char(10)
     FIELD SCRIPT_NO numeric(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 = 'P' V T.OUT_ROUTING
                        = 'F')V
                       (T.OUT_FILE != null ∧ T.OUT_ROUTING
                        = 'F'))

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 ACCESS_TYPE char(10)
     FIELD USER_CLASS char(20))
  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'  V 'DBA'
                        V'DELETE' V 'DISTAPE'  V'FORM'
                        V'GENERAL' V  'IMPORT'  V 'INSERT'
                        V'QA' V  'QUERY' V  'REPORT'
                        V'RESTORE' V  'UPDATE'  V 'VIEW'))
```

```
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(6)
      FIELD T_OLD_LINE numeric(6)
      FIELD T_MOD_LINE numeric(6)
      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)

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 VAL_MEAS_TYPE
    (FIELD CODE,SOURCE VALIDATION
     FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_SECOND_L
    (FIELD CODE,SOURCE VALIDATION
     FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_ACTIVE_STATUS
         (FIELD CODE,SOURCE VALIDATION)
         (FIELD CODE,SOURCE VALIDATION)

VIEW VAL_MESS_TYPE
    (FIELD CODE,SOURCE VALIDATION
     FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_STATUS
    (FIELD CODE,SOURCE VALIDATION
     FIELD VALUE,SOURCE VALIDATION)
```

```
VIEW VAL_S_FUNCTION
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_COM_PURPOSE
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_ORI_TYPE
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_COM_TYPE
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_ADA_FEATURE
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_ERR_CLASS
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_CH_TYPE
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_ERR_ARES
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_ERR_SOURCE
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_ERR_ACAUSE
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_ERR_TOOLS
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_ACTIVITY
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)
```

```
VIEW V_PROJ_TYPE
   (FIELD PROJ NO,SOURCE PROJECT
    FIELD PROJ TYPE,SOURCE PROJECT)

VIEW VAL_PHASE_CO
   (FIELD CODE,SOURCE VALIDATION
    FIELD VALUE,SOURCE VALIDATION)

VIEW V_PERM_SCRIPT
   (FIELD SCRIPT_NAME,SOURCE PERM_SCRIPT)

VIEW V_REP_CODES_CRITERIA
   (FIELD VALUE,SOURCE REP_CODES)

VIEW VAL_COM_CH
    (FIELD CODE,SOURCE VALIDATION
     FIELD VALUE,SOURCE VALIDATION)

VIEW VAL_ISO_CH
    (FIELD CODE,SOURCE VALIDATION
     FIELD VALUE,SOURCE VALIDATION)

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_SP_ACTIVITY
    (FIELD CODE,SOURCE VALIDATION
     FIELD VALUE,SOURCE VALIDATION)

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)

VIEW V_PERM_SCRIPT
      (FIELD SCRIPT_NAME,SOURCE PERM_SCRIPT)

VIEW V_REP_CODES_LOG
      (FIELD VALUE,SOURCE REP_CODES)
```

REFERENCES

1. Software Engineering Laboratory, SEL-87-008, Data Collection Procedures for the Rehosted SEL Database, G. Heller, October 1987

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, User's Guide for the Database Access Manager for the Software Engineering Laboratory (DAMSEL), S. Steinberg, April 1989

4. ORACLE Corporation, SQL*Plus User's Guide, J. Sachs

5. ORACLE Corporation, SQL*Plus Reference Guide, J. Sachs

6. C. J. Date, An Introduction to Database Systems, Addison Wesley

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 and W. A. Taylor, 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

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-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 From 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, W. J. Decker, W. A. Taylor, and E. J. Smith, 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-81-205, Recommended Approach to Software Development, F. E. McGarry, G. Page, S. Eslinger, et al., April 1983

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 From 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, F. E. McGarry, and M. G. Rohleder, October 1983

SEL-82-706, Annotated Bibliography of Software Engineering Laboratory Literature, G. Heller, January 1989

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 From the Eighth Annual Software Engineering Workshop, November 1983

SEL-84-001, Manager's Handbook for Software Development, W. W. Agresti, F. E. McGarry, D. N. Card, et al., April 1984

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 From the Ninth Annual Software Engineering Workshop, November 1984

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., May 1985

SEL-85-005, Software Verification and Testing, D. N. Card, C. Antle, and E. Edwards, December 1985

SEL-85-006, Proceedings From 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 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, October 1986

SEL-86-006, Proceedings From 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-87-008, Data Collection Procedures for the Rehosted SEL Database, G. Heller, October 1987

SEL-87-009, Collected Software Engineering Papers: Volume V, S. DeLong, November 1987

SEL-87-010, Proceedings From the Twelfth Annual Software Engineering Workshop, December 1987

SEL-88-001, System Testing of a Production Ada Project: The GRODY Study, J. Seigle 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-89-001, Software Engineering Laboratory (SEL) Database Organization and User's Guide, M. So, G. Heller, S. Steinberg, and D. Spiegel, May 1989

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

[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

[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

[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

Basili, V. R., and J. Ramsey, Structural Coverage of Functional Testing, University of Maryland, Technical Report TR-1442, September 1984

[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 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. 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. and H. D. Rombach, "T A M E: Tailoring an Ada Measurement Environment," Proceedings of the Joint Ada Conference, March 1987

[5]Basili, V. 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

[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

[3]Basili, V. R., and R. W. Selby, Jr., "Calculation and Use of an Environments'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 "Four Applications of a Software Data Collection and Analysis Methodology," Proceedings of the NATO Advanced Study Institute, August 1985

[4]Basili, V. R., R. W. Selby, Jr., and D. H. Hutchens, "Experimentation in Software Engineering," IEEE Transactions on Software Engineering, July 1986

[5]Basili, V. and R. Selby, "Comparing the Effectiveness of Software Testing Strategies," IEEE Transactions on Software Engineering, December 1987

[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

[5]Brophy, C., W. Agresti, and V. 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. and W. Agresti, "Resolving the Software Science Anomaly," The Journal of Systems and Software, 1987

[6]Card, D. N., and W. Agresti, "Measuring Software Design Complexity," The Journal of Systems and Software, June 1988

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

[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., Q. L. Jordan, and V. E. Church, "Characteristics of FORTRAN Modules," Computer Sciences Corporation, Technical Memorandum, June 1984

[5]Card, D., F. McGarry, and G. 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

[5]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)

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. and 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

[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. 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

[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

[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

[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

[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

Stark, M., and E. Seidewitz, "Towards a General Object-Oriented Ada Lifecycle," Proceedings of the Joint Ada Conference, March 1987

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. and F. 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. 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

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.